Федеральное государственное бюджетное учреждение науки
Институт проблем управления им. В.А. Трапезникова
РОССИЙСКОЙ АКАДЕМИИ НАУК

А.А. Лазарев

ТЕОРИЯ РАСПИСАНИЙ

МЕТОДЫ И АЛГОРИТМЫ

Москва
ИПУ РАН
2019

УДК 519.854.2
ББК 22.18
Л17

Лазарев А.А. Теория расписаний. Методы и алгоритмы / А.А. Лазарев. — М. : ИПУ РАН, 2019. — 408 с. — ISBN 978-5-91450-236-9.

Монография продолжает серию книг по теории расписаний и является результатом многолетних научных исследований автора. В книге рассматриваются фундаментальные задачи теории расписаний с минимаксными и суммарными критериями. Введено понятие метрики для задач теории расписаний. Сформулированы, обоснованы и построены алгоритмы нахождения приближенных решений задач теории расписаний с минимальной гарантированной абсолютной погрешностью целевой функции и предложена новая схема нахождения приближённого решения данных задач для нахождения эффективных нижних оценок целевой функции, которые можно использовать в методах сокращённого перебора поиска оптимального решения задачи.

Научное издание предназначено для специалистов в области дискретной оптимизации, а также для аспирантов и студентов математических специальностей.

Рецензенты: д.т.н., проф. Ф.Т. Алескеров,
д.т.н., проф. В.Н. Бурков

Утверждено к печати Редакционным советом Института

Работа выполнена при финансовой поддержке Российского научного фонда (проект № 17—19—01665)

ISBN 978-5-91450-236-9

Оглавление

Введение

Общее направление исследований

Люди на протяжении всей жизни сталкиваются с проблемами составления расписаний. В обычной жизни эти проблемы решаются интуитивно. Но даже на обыденном уровне человек исполняет алгоритмы, пусть даже не осознавая этого. Часто мы планируем наши действия в порядке возрастания крайних сроков исполнения работ. Например, студенты во время экзаменационной сессии учат предмет с наименьшим директивным сроком (ближайший по дате сдачи экзамен), тем самым они минимизируют максимальное временно́е смещение. Так как лучше получить на экзаменах четыре четвёрки, чем три пятерки и одну тройку,— стипендии не будет... Для решения бытовых вопросов применение интуитивного подхода оказывается достаточно.Человек при составлении расписания для себя фактически реализует некоторым образом (как он думает,— наилучшим для себя) целевую функцию. Можно сказать: скажи мне свое расписание и я скажу кто ты...

Расписания, по которым мы живем, являются плодом нашего сознания, воспитания (или его отсутствия), образования, внешних обстоятельств. В свою очередь расписания формируют наше бытие.

Теория расписаний наглядно показывает одно из противоречий, которое будет существовать в человеческом обществе до тех пор, пока в нем будут существовать организационные структуры. Исполнитель, в подавляющем большинстве случаев, принимает решение по минимаксному критерию (или максиминному критерию в другой интерпретации), т.е. исполнитель пытается получить максимум прибыли по каждому из проектов, над которыми работает. Структура (руководство лаборатории, цеха; дирекция института, завода; руководитель министерства и т.д.), в подавляющем большинстве случаев, руководствуется суммарным критерием при выполнении всей совокупности проектов, стоящих перед ней. И здесь возникает конфликт интересов... Необходимо заметить, когда руководство лаборатории (цеха, бригады и т.д.) выступает в роли "исполнителя" перед руководством института (завода и т.д., т.е. следующий уровень организационной структуры), то оно начинает

действовать по минимаксному критерию.

То есть в организационных структурах, которые мы рассматриваем, каждый предшествующий уровень структуры руководствуется минимаксным критерием при взаимодействии с вышестоящими уровнями структуры. В то время как, вышестоящие уровни структуры руководствуются суммарными критериями при работе с нижестоящими уровнями структуры.

В данной работе мы как раз и исследуем природу минимаксного и суммарного критериев при принятии решений.

Заметим, что при принятии решений присутствуют как рациональная составляющая, так и иррациональная (бескультурье, человеческие пороки и т.п.) составляющая, которая выходит за рамки настоящего исследования.

Развивающаяся стремительными темпами автоматизация производства и неуклонно увеличивающиеся его масштабы ставят перед научным сообществом всё более и более трудные задачи разработки алгоритмов составления расписаний.

Теория расписаний является одним из разделов исследования операций. Термин "теория расписаний" предложил Р. Беллман в 1956 году (Bellman [87]). Методы и алгоритмы решения задач теории расписаний применяются для решения практических задач комбинаторной и дискретной оптимизации.

Данное направление в науке, получившее название *"теория расписаний"*, берёт свое начало с известной работы в данной области Генри Гантта 1903 г. (Gantt H.L. [121]), предложившего то, что сегодня называют диаграммами Гантта, которые встречаются во многих работах по теории расписаний, в том числе и в данной работе. С 50-х годов 20-го века началось активное теоретическое исследование задач теории расписаний. Здесь следует отметить работы Джонсона (Johnson [135]), Джексона (Jackson [133]) и Смита (Smith [204]), а также монографии [13, 63].

Одним из первых вопросов нового направления была классификация задач и установление их сложности. В настоящее время наиболее устоявшаяся классификация задач теории расписаний была предложена Грэхэмом и др. (Graham at al. [125]). Относительно быстро была установлена сложность большого числа задач. Достаточно полные обзоры по задачам теории расписаний и их сложности представлены в работах Гэри и Джонсона (Garey, Johnson [7]), Ленстры и др. (Lenstra et al. [167]), Лоулера и др. (Lawler et al. [147]), Танаева и др. [63, 64, 65], Брукера и др. [98, 99].

Рассмотрим общую постановку задачи теории расписаний. Множество требований (работ) $N = \{1, 2, \ldots, n\}$ должно быть обслужено без (или с) пре-

рываний на одном или нескольких приборах $M_i, i = 1, \ldots, m$, которые могут обслуживать не более одного требования в каждый момент времени. Время обслуживания требования $j \in N$ на приборе M_i обозначается как $p_{ji}, p_{ji} > 0$. Момент, когда требование j становится доступным для обслуживания, задаётся временем поступления r_j. Требование j может иметь директивный срок d_j (время, до которого желательно завершить обслуживание требования), а также вес w_j (значимость, важность требования). Между требованиями могут быть заданы отношения предшествования в виде ациклического ориентированного графа G. В большинстве задач теории расписаний целью является построение оптимального расписания по определённому критерию или совокупности критериев. В практических задачах теории расписаний обычно для обслуживания требований необходимы один или несколько ресурсов. Для представления различных критериев нам необходимы следующие определения.

Момент окончания обслуживания требования j при расписании π будем обозначать через $C_j(\pi)$. Определим $L_j(\pi) = C_j(\pi) - d_j$ как *временно́е смещение* требования j при расписании π, а $T_j(\pi) = \max\{0, L_j(\pi)\}$ — как его *запаздывание*. $U_j(\pi)$ обозначает запаздывает ли требование j при расписании π или нет: $U_j(\pi) = 1$, если j *запаздывает* ($C_j(\pi) > d_j$), иначе $U_j(\pi) = 0$ (требование j *обслуживается вовремя*). Классическими критериями в задачах теории расписаний являются: $C_{\max}$ — минимизация максимального времени окончания обслуживания всех требований (задача на быстродействие)[1]; $L_{\max}$ — минимизация максимального временно́го смещения; $\sum T_j$ — минимизация суммарного запаздывания; $\sum w_j U_j$ — минимизация взвешенного числа запаздывающих требований. Заметим, что данные критерии являются *регулярными*, т.е. данные функции являются неубывающими по отношению к моментам окончания обслуживания требований. При наличии регулярного критерия мы можем ограничиться рассмотрением только *ранних расписаний*, без искусственных простоев приборов.

Через π_i будем обозначать перестановку (расписание) из элементов множества N, обслуживаемых на приборе $M_i, i = 1, \ldots, m$, множество требований в расписании π_i обозначим $\{\pi_i\} \subseteq N$, а через $\pi = \bigcup_{i=1}^{m} \pi_i$, $\{\pi_\alpha\} \bigcap \{\pi_\beta\} = \emptyset$, если $\alpha \neq \beta$, расписание для всего множества требований $N = \{\pi\}$. Рассматриваются только *допустимые* расписания, удовлетворяющие отношениям предшествования и моментам поступления требований r_j, $j \in N$. Для любого ациклического графа G множество допустимых расписаний не пусто. Момент окончания обслуживания требования j на приборе M_i при расписа-

[1] в англоязычной литературе *makespan*

нии π определяется как

$$C_j(\pi) = \max\left\{r_j, \max_{k\in N_j} C_k(\pi), \max_{(k\to j)_{\pi_i}} C_k(\pi_i)\right\} + p_{ji},$$

где N_j — множество требований предшествующих требованию j, согласно графу G, и $(k \to j)_{\pi_i}$ требования, согласно расписанию π_i на приборе M_i, предшествующих обслуживанию требования $j \in \{\pi_i\}$. Предполагается, что для пустого расписания $C_k(\pi) = r_k + p_k$ для всех требований k, для которых нет предшествующих требований, при обслуживании на приборе M_i. Множество расписаний обслуживания требований множества N, допустимых относительно графа предшествования G, будем обозначать через $\Pi(N)$.

Если обслуживание требования i предшествует обслуживанию требования j, где $i, j \in N$, при расписании π, то будем обозначать через $(i \to j)_\pi$ или $i \xrightarrow{\pi} j$, чтобы "не перегружать" нижний индекс.

Для представления задач построения оптимальных расписаний обслуживания требований множества N, рассмотренных в данной работе, мы будем использовать обозначение $\alpha|\beta|\gamma$ [125]. Первое поле α описывает обслуживающую систему: P – параллельные идентичные приборы; Q – параллельные приборы разной производительности; R – параллельные различные приборы; F – $flow-shop$ – каждое требование должно быть обслужено каждым из приборов множества $M = \{1, \ldots, m\}$ в порядке $1, \ldots, m$. В русскоязычной литературе данный класс задач обычно называют задачами конвейерного типа. O – $open-shop$ – порядок обслуживания требования на приборах может быть произвольным; J – $job-shop$ – у каждого требования свой технологический порядок обслуживания. Во втором поле β описываются характеристики требований и отношения предшествования между ними: r_j – заданы моменты поступления требований на обслуживание; $p_j = p$ – все требования имеют одинаковые продолжительности обслуживания; $prec$ – между требованиями заданы отношения предшествования; $pmtn$ – разрешены прерывания при обслуживании требований. Третье поле γ описывает критерий задачи, который состоит в отыскании допустимого (относительно графа предшествования) расписания, минимизирующего (или максимизирующего) *целевой функционал*.

Отметим, что подавляющее большинство исследованных задач теории расписаний являются NP-трудными. Несмотря на это, практика требует решение таких задач. Для этого существует несколько подходов.

Первым подходом является разработка полиномиальных эвристических алгоритмов. Для многих эвристических алгоритмов были найдены оценки

погрешности получаемого решения. Такие алгоритмы называются *приближёнными* [16, 58]. Существуют приближённые алгоритмы, гарантирующие как относительную погрешность [9, 198], так и абсолютную погрешность [61]. Некоторые NP-трудные задачи допускают существование так называемой *аппроксимационной схемы*. В рамках данной схемы можно найти приближённое решение с относительной погрешностью не более любого заданного значения $\varepsilon > 0$ за время, полиномиально зависящее от $1/\varepsilon$ и от размера входной информации задачи,— вполне полиномиальная аппроксимационная схема ($FPTAS$). Для задач теории расписаний такие схемы разработали, например, Ковалёв [12], Алон и др. (Alon et al. [74]), Мастролилли (Mastrolilli [172]), Севастьянов и Вёгингер (Sevastianov, Woeginger [199]). Для задач, не имеющих аппроксимационной схемы, большое значение имеет установление предельного значения ε, для которого возможно нахождения ε-приближённого решения за полиномиальное время,— полиномиальная аппроксимационная схема ($PTAS$).

В настоящий момент широкое распространение имеют метаэвристические алгоритмы (еще их называют алгоритмами “локального поиска”), которые находят “хорошее” решение, близкое к оптимальному, за приемлемое время. Недостатком таких алгоритмов является отсутствие оценок качества полученного решения. Неизвестно, насколько решение отличается от оптимального в наихудшем случае.

Точным методам решения NP-сложных задач также уделено немалое внимание в работах по теории расписаний. Наибольшее распространение получили методы сокращённого перебора, также называемые методами ветвей и границ [14, 62, 93, 101, 142]. Для сокращения перебора вычисляются нижние оценки целевой функции (в случае её минимизации) и используются комбинаторные свойства задач. Также для решения задач теории расписаний широко применяется метод динамического программирования [1, 7, 58, 59, 98, 137].

Часто задачи теории расписаний могут быть сформулированы как задачи целочисленного линейного программирования. Решению таких задач посвящены, например, работы [71, 72, 189, 205].

В последнее время широкое распространение получил метод программирования в ограничениях (ПвО, в англоязычной литературе – Constraint Programming). Одной из областей его успешного применения является теория расписаний [84].

Некоторые сложные задачи теории расписаний могут быть оптимально решены с помощью алгоритмов, использующих элементы сразу нескольких методов. Одно из их названий — “гибридные алгоритмы” [15, 134].

Научная новизна

Для задач теории расписаний для одного и нескольких приборов с критерием минимизации максимального временно́го смещения впервые введена метрика ρ. Доказана теорема об абсолютной погрешности.

Предложена новая схема нахождения приближённого решения задач теории расписаний, состоящая из двух этапов. На первом этапе, решая задачу линейного программирования, находится такое изменение исходных параметров примера A $(r_j^A, p_{ji}^A, d_j^A | j \in N, i \in M)$, чтобы полученный пример B с параметрами требований $(r_j^B, p_{ji}^B, d_j^B | j \in N, i \in M)$ принадлежал заданному полиномиально/псевдополиномиально разрешимому классу примеров исходной NP-трудной задачи и был на минимальном расстоянии от примера A в метрике ρ. На втором этапе для решения примера B используется уже известный для данного класса примеров полиномиальный/псевдополиномиальный алгоритм, а затем найденную им оптимальную перестановку требований применим к исходному примеру A. Абсолютная погрешность такого решения не будет превосходить расстояния $\rho(A, B)$ между примерами. Данный подход позволяет находить эффективные нижние оценки целевой функции, которые можно использовать в методах сокращённого перебора для оптимального решения задачи.

Поставлена и решена задача, в некотором смысле "двойственная"к задаче $1 \mid r_j \mid f_{\max}$ при произвольных неубывающих функциях штрафа, трудоёмкость алгоритма $O(n^2)$ операций. Решение данной задачи является оценкой снизу для исходной NP-трудной задачи $1 \mid r_j \mid f_{\max}$ и также может быть использована в алгоритмах сокращённого перебора, в частности, в методе ветвей и границ.

Для NP-трудной в обычном смысле задачи $1 \mid\mid \sum T_j$, когда параметры требований удовлетворяют ограничениям $p_1 \geq p_2 \geq \ldots \geq p_n$, $d_1 \leq d_2 \leq \ldots \leq d_n$, построен псевдополиномиальный алгоритм трудоёмкости $O(n^2 \sum p_j)$ операций.

Предложена графическая реализация метода динамического программирования, на основе которой построены алгоритмы решения задач РАЗБИЕНИЕ и РАНЕЦ, показавшие лучшую экспериментальную эффективность по сравнению с представленными в научной литературе. Алгоритмы позволяют решать примеры с нецелочисленными и отрицательными значениями параметров.

Структура и обзор результатов

В разделе 1.1 **первой главы** рассматривается постановка задачи минимизации максимального временно́го смещения ($L_{\max}$), дан краткий исторический обзор задач, приведены основные результаты, существующие в литературе. В разделе 1.2 вводятся обозначения и определения, которые используются далее в работе. В разделе 1.3 формулируется и доказывается теорема об абсолютной погрешности [29].

Теорема 0.1 *Пусть* $A = \{G, (r_j^A, p_j^A, d_j^A) | \, j \in N\}$ *и* $B = \{G, (r_j^B, p_j^B, d_j^B) | \, j \in N\}$ *(с идентичным графом предшествования* G*) –два примера* NP*-трудной задачи* $P \mid prec, r_j \mid L_{\max}$*, тогда*

$$0 \leq L_{\max}^A(\pi^B) - L_{\max}^A(\pi^A) \leq \rho(A, B),$$

где π^A, π^B *– оптимальные расписания примеров* A *и* B*,*

$$\rho(A, B) = \rho_r(A, B) + \rho_d(A, B) + \rho_p(A, B),$$

$$\rho_r(A, B) = \max_{j \in N}\{r_j^A - r_j^B\} - \min_{j \in N}\{r_j^A - r_j^B\},$$
$$\rho_d(A, B) = \max_{j \in N}\{d_j^A - d_j^B\} - \min_{j \in N}\{d_j^A - d_j^B\},$$
$$\rho_p(A, B) = \sum_{j \in N} |p_j^A - p_j^B|.$$

Мы можем использовать оптимальное расписание примера B для решения исходного примера A. Если пример B принадлежит некоторому полиномиально/псевдополиномиально разрешимому подмножеству примеров, то необходимо найти такой пример из этого подмножества, до которого расстояние $\rho(A, B)$ было бы минимальным. Тогда искомое оптимальное значение целевой функции будет очевидно принадлежать интервалу $L_{\max}^A(\pi^A) \in$
$\in [L_{\max}^A(\pi^B) - \rho(A, B), L_{\max}^A(\pi^B)]$.

Рассмотрим полиномиально/псевдополиномиально разрешимый случай исходной задачи, когда параметры требований удовлетворяют k линейно независимым неравенствам

$$XR + YP + ZD \leq H,$$

где $R = (r_1, \ldots, r_n)^T$, $P = (p_1, \ldots, p_n)^T$, $D = (d_1, \ldots, d_n)^T$, и X, Y, Z – матрицы размерности $k \times n$, а $H = (h_1, \ldots, h_k)^T$ – k-мерный вектор (верхний индекс ^{T} обозначает операцию транспонирования). Затем в этом классе примеров находим пример B с минимальным расстоянием $\rho(A, B)$ до исходного примера A, решая следующую задачу

$$\begin{cases} (x^d - y^d + x^r - y^r) + \sum\limits_{j \in N} x_j^p \longrightarrow \min \\ y^d \le d_j^A - d_j^B \le x^d, \quad j = 1, \ldots, n, \\ y^r \le r_j^A - r_j^B \le x^r, \quad j = 1, \ldots, n, \\ -x_j^p \le p_j^A - p_j^B \le x_j^p, \quad j = 1, \ldots, n, \\ 0 \le x_j^p, \quad j = 1, \ldots, n, \\ XR^B + YP^B + ZD^B \le H. \end{cases}$$

Данная задача линейного программирования с $3n+4+n$ переменными ($r_j^B,\ p_j^B,\ d_j^B,\ j = 1, \ldots, n,$ и $x_d,\ y_d,\ x_r,\ y_r,$ и $x_j^p,\ j = 1, \ldots, n$) и $7n + k$ неравенствами иногда может быть решена за полиномиальное (от n) операций, учитывая специфику ограничений задачи. Для всех известных полиномиально/псевдополиномиально разрешимых случаев задачи $P \mid prec, r_j \mid L_{\max}$ количество ограничений $k = O(n)$.

Рассмотрим совокупность примеров задачи $P|prec, r_i|L_{\max}$ с количеством требований n, количеством приборов m и ориентированным ациклическим графом предшествования G. Данная совокупность примеров образует $3n$-мерное линейное пространство ($3n$ параметров: $r_j, p_j, d_j, j = \overline{1, n}$). Два примера $A = \{G, (r_j^A, p_j^A, d_j^A) | \, j \in N\}$ и $B = \{G, (r_j^B, p_j^B, d_j^B) | \, j \in N\}$ будем называть эквивалентными, если существуют константы d и r, что $d_j^A = d_j^B + d,\ r_j^A = r_j^B + r,\ p_j^A = p_j^B,\ \forall j \in N$. Полученное семейство классов эквивалентности является $(3n-2)$-мерным линейным пространством, которое обозначим через $\mathcal{L}_n$. Будем говорить, что пример A принадлежит классу $\mathcal{L}_n$, если выполняется условие $r_1 = d_1 = 0$. Несложно показать, что у эквивалентных примеров совпадают множества оптимальных расписаний. На пространстве классов эквивалентности примеров рассмотрим следующий функционал

$$\forall A \in \mathcal{L}_n, \quad \varphi(A) = \max_{j \in N} r_j^A - \min_{j \in N} r_j^A + \max_{j \in N} d_j^A - \min_{j \in N} d_j^A + \sum_{j \in N} |p_j^A| \ge 0.$$

Данный функционал удовлетворяет трём свойствам нормы:

$$\varphi(A) = 0 \Longleftrightarrow A \equiv 0; \varphi(\alpha A) = \alpha \varphi(A); \varphi(A + B) \le \varphi(A) + \varphi(B).$$

Таким образом, $\mathcal{L}_n$ является $(3n-2)$-мерным линейным нормированным пространством с нормой $||A|| = \varphi(A)$. Данная норма естественным образом порождает метрику $\rho(A, B) = ||A - B||$. Необходимо заметить, что для функции $\rho(A, B)$ верна следующая теорема.

Теорема 0.2 *[29] Функция $\rho(A, B) = ||A - B||$ удовлетворяет свойствам нормированной метрики в $(3n-2)$-мерном линейном пространстве параметров требований $\{(r_j, p_j, d_j) | j \in N\}$.*

В случае, когда для исходной задачи нет полиномиально и псевдополиномиально разрешимых выделенных подслучаев задачи (в скобках заметим, что тривиальные подслучаи обычно не позволяют найти качественно новые оценки абсолютной погрешности), либо "расстояние" $\rho(A, C)$ до любого полиномиально/псевдополиномиально разрешимого примера C "слишком велико", тогда можно использовать следующую теорему.

Теорема 0.3 *Пусть* $A = \{G, (r_j^A, p_j^A, d_j^A) | \, j \in N\}$ *и* $B = \{G, (r_j^B, p_j^B, d_j^B) | \, j \in N\}$ (*с идентичным графом предшествования* G) *два примера, тогда для любого приближённого расписания* $\overline{\pi}$ *верно*

$$0 \leq L_{\max}^A(\overline{\pi}) - L_{\max}^A(\pi^A) \leq \delta^B(\overline{\pi}) + \rho(A, B),$$

где $\delta^B(\overline{\pi}) = L_{\max}^B(\overline{\pi}) - L_{\max}^B(\pi^B)$ *погрешность приближённого решения примера* B.

Два параметра необходимо "фиксировать", так как множества оптимальных расписаний для примеров $\{(r_j, p_j, d_j) | j \in N\}$ и $\{(r_j + a, p_j, d_j + b) | j \in N\}$ совпадают для любых $a, b \in R$. Например, можно положить $a = -r_1, b = -d_1$, что и было сделано при определении класса $\mathcal{L}_n$. Функцию $\rho(A, C)$ можно рассматривать, как *расстояние* между произвольно выбранными примерами A и C.

Предлагается общая схема приближённого решения задач минимизации $L_{\max}$. Идея предлагаемого подхода состоит в построении по исходному примеру A задачи другого примера, для которого удаётся найти оптимальное или приближённое решение, с минимальным расстоянием до исходного примера в метрике $\varphi(A)$.

Во **второй главе** представлены новые полиномиально и псевдополиномиально разрешимые случаи NP-трудной в сильном смысле задачи $1|r_j|L_{\max}$. В разделе 2.1 рассмотрен случай [24, 28, 49], когда параметры требований удовлетворяют следующим ограничениям:

$$\begin{cases} d_1 \leq \ldots \leq d_n; \\ d_1 - r_1 - p_1 \geq \ldots \geq d_n - r_n - p_n. \end{cases}$$

Данным ограничениям соответствует случай, когда $d_j = r_j + p_j + z$, $j = 1, \ldots, n$, где z – константа, т.е. когда все требования имеют одинаковый запас времени до своего директивного срока. Алгоритмом [28] за $O(n^3 \log n)$ операций может быть найдено Парето-оптимальное множество (по критериям $L_{\max}$ и $C_{\max}$), состоящее не более чем из n расписаний, решения общей двухкритериальной задачи $1|r_j|L_{\max}, C_{\max}$.

Теорема 0.4 *[48, 49] Для любого примера A задачи $1 \mid r_j \mid L_{\max}$, не принадлежащего исследуемому классу, и для любого примера C, наследующего длительности обслуживания примера A и принадлежащего данному классу, справедлива оценка расстояния между A и C:*

$$\rho(A,C) \geq \rho^L(A) = \max_{i,j \in N} \min\{d_j^A - d_i^A, (d_j^A - r_j^A - p_j^A) - (d_i^A - r_i^A - p_i^A)\}.$$

Оценка достигается на некотором примере C, который может быть найден за время $O(n \log n)$ операций.

Кроме того, был рассмотрен полиномиально разрешимый случай Хогевена [130]: $\max\limits_{k \in N}\{d_k - r_k - p_k\} \leq d_j - r_j$, $\forall j \in N$. Оптимальное расписание находится за $O(n^2 \log n)$ операций.

Теорема 0.5 *Для любого примера A задачи $1|r_j|L_{\max}$, не принадлежащего классу Хогевена, и для любого примера C, наследующего длительности обслуживания примера A и принадлежащего классу Хогевена, справедлива оценка расстояния между A и C:*

$$\rho(A,C) \geq \rho^H(A) = \max_{i,j \in N}\{d_j^A - r_j^A - p_j^A - d_i^A + r_i^A\}.$$

Оценка достигается на некотором примере C, который может быть найден за время $O(n)$ операций.

Для более сложного NP-трудного подслучая задачи $1|r_j|L_{\max}$, когда параметры требований удовлетворяют ограничениям:

$$\begin{cases} d_1 \leq d_2 \leq \ldots \leq d_n; \\ r_1 \geq r_2 \geq \ldots \geq r_n; \\ r_j, p_j, d_j \in \mathbb{Z}^+, \forall j \in N, \end{cases}$$

предлагается псевдополиномиальный алгоритм решения [54, 57] трудоёмкости $O(n^2P + np_{\max}P)$ операций, где $P = \max\{r_{\max}, t\} + \sum\limits_{j=1}^{n} p_j - \max\{r_{\min}, t\}$, $r_{\max} = \max\limits_{j \in N} r_j$, $r_{\min} = \min\limits_{j \in N} r_j$, $p_{\max} = \max\limits_{j \in N} p_j$, а t– момент времени, с которого прибор готов начать обслуживать требования множества N.

Для получения нижних оценок целевой функции был также использован следующий подход. Рассмотрим общую постановку NP-трудной задачи $1|r_j|f_{\max}$:

$$\mu^* = \min_{\pi \in \Pi(N)} \max_{k=\overline{1,n}} f_{j_k}(C_{j_k}(\pi)),$$

где $f_{j_k}(C_{j_k}(\pi))$– произвольные неубывающие функции штрафа. Решим задачу нахождения величины ν^*:

$$\nu^* = \max_{k=\overline{1,n}} \min_{\pi \in \Pi(N)} f_{j_k}(C_{j_k}(\pi)).$$

Теорема 0.6 *[19, 24] Для NP-трудной задачи $1|r_j|f_{\max}$ с произвольными неубывающими функциями штрафа $f_j(t), \forall j \in N$, верно $\nu^* \leq \mu^*$ и величина ν^* может быть найдена за $O(n^2)$ операций.*

Величина ν^* может быть использована как нижняя граница в алгоритме ветвей и границ решения задачи $1|r_j|f_{\max}$.

Необходимо заметить, что для данной проблемы теории расписаний исходная задача является NP—трудной в сильном смысле, а двойственная к ней — полиноминально разрешима.

Третья глава посвящена алгоритмам оптимального решения задачи $1 \mid r_j \mid L_{\max}$. В разделе 3.1 рассмотрены существующие подходы к решению задачи, такие как алгоритм Карлье [101] и метод программирования в ограничениях [84] (ПвО). Данный метод близок к методу ветвей и границ. Отличие заключается в том, что для сокращения перебора в ПвО используется пропагация ограничений, которая удаляет несовместимые значения из множеств допустимых значений переменных.

В разделе 3.2 предлагаются два алгоритма оптимального решения задачи $1 \mid r_j \mid L_{\max}$. Первый алгоритм построен по методу ПвО. На каждом шаге алгоритма для исходной задачи решается задача распознавания с помощью метода ПвО, в котором используется предложенное нами правило ветвления, основанное на следующей теореме.

Теорема 0.7 *Пусть построено расписание Шраге [192] π^σ и требования пронумерованы в том порядке, в котором они упорядочены в π^σ. Пусть $b \in N$ – наименьший номер, для которого $L_b(\pi^\sigma) = L_{\max}(\pi^\sigma)$, и $a \in N$ – наибольший такой номер, что $a \leq b$ и*

$$C_a(\pi^\sigma) - p_a - \min_{a \leq j \leq b} r_j \leq L_b(\pi^\sigma) - UB,$$

где $UB \leq L_{\max}(\pi^\sigma)$, UB – верхняя оценка целевой функции оптимального решения. Примем $S = \{a, \ldots, b\}$, тогда:

- *если не существует такого требования $c \in S$, что $d_c > d_b$, то не существует расписания с $L_{\max}$ меньшим, чем UB;*

- *если $c \in S$ – последнее требование с директивным сроком $d_c > d_b$, то в любом расписании π', что $L_{\max}(\pi') < UB$, выполняется либо $(c \to J)_{\pi'}$, либо $(J \to c)_{\pi'}$, где $J = \{c+1, \ldots, b\}$.*

Расписание Шраге: на каждое последующее место в расписание ставится требование с минимальным директивным сроком среди уже поступивших на обслуживание; если таких нет, то выбирается требование среди имеющих минимальное время поступления среди неупорядоченных.

Второй алгоритм построен по методу ветвей и границ. В алгоритме также используется правило ветвления, основанное на теореме 0.7. Отличительной особенностью алгоритма является использование алгоритмов ПвО на каждом узле дерева поиска.

В разделе 3.3 приводятся результаты экспериментального сравнения точных алгоритмов решения задачи $1 \mid r_j \mid L_{\max}$, предложенных в работе, и подходов, существующих в литературе.

В разделе 3.4 рассматривается вариант распознавания задачи $1 \mid r_j \mid L_{\max}$. Назовем расписание π *допустимым* по отношению к константе L', если $L_{\max}(\pi) \leq L'$. Множество требований S *допустимо* по отношению к константе L', если существует допустимое расписание $\pi \in \Pi(S)$, и *недопустимо*, если не существует такого расписания. В рассматриваемом варианте задачи требуется определить допустимо ли множество требований N по отношению к заданной константе L'. Если N допустимо, то необходимо найти допустимое расписание. Если же N недопустимо, то требуется найти как можно меньшее недопустимое подмножество требований из множества N. Для данного варианта задачи предлагается алгоритм, который является эвристическим в том смысле, что он находит некоторое недопустимое подмножество требований $S \subseteq N$, не обязательно минимальное, когда множество требование N недопустимо. В тоже время, алгоритм точно определяет, допустимо ли множество требование N. Правило ветвления алгоритма основано на следующей теореме.

Теорема 0.8 *Пусть $L_{\max}(\pi^\sigma) > L'$ и требования пронумерованы в том порядке, в котором они упорядочены в π^σ (алгоритм Шраге [192]), $b \in N$ – наименьший номер, для которого $L_b(\pi^\sigma) > L'$, и $a \in N$ – наибольший номер, что $a \leq b$ и*

$$C_a(\pi^\sigma) - p_a - \min_{j \in S} r_j < L_b(\pi^\sigma) - L',$$

где $S = \{a, \ldots, b\}$. Тогда:

- *если не существует такого требования $c \in S$, что $d_c > d_b$, то множество требований S недопустимо по отношению к L';*
- *если $c \in S$ – последнее требование с директивным сроком $d_c > d_b$, и существует допустимое расписание π', то выполняется либо $(c \to J)_{\pi'}$, либо $(J \to c)_{\pi'}$, где $J = \{c+1, \ldots, b\}$.*

Отличительной чертой модифицированного алгоритма Карлье является способ построения недопустимого подмножества требований в случае, если алгоритм не нашёл допустимого расписания. На каждом узле дерева поиска, где не происходит ветвления, согласно теореме 0.8, мы располагаем подмножеством S, недопустимым для задачи с текущими параметрами требований. Данное подмножество "передаётся родительскому" узлу дерева поиска. Способ построения недопустимого подмножества для узла, имеющего "дочерние" узлы, обоснован в следующей теореме.

Теорема 0.9 *Пусть при любом допустимом расписании $\pi' \in \Pi(N)$ требование $c \in N$ обслуживается до или после всех требований множества $J \subset N$, $S' \subset N$ – недопустимое множество требований, когда $(c \to J)_{\pi'}$ а $S'' \subset N$ – недопустимое подмножество, когда $(J \to c)_{\pi'}$. Тогда:*

- *если $c \notin S'$ $(c \notin S'')$, то множество S' (S'') недопустимо;*
- *если $c \in S'$ и $c \in S''$, то множество $S = J \cup S' \cup S''$ недопустимо.*

В **четвёртой главе** предлагается точный алгоритм решения задачи $1 \mid r_j \mid \sum w_j U_j$. Алгоритм построен по "гибридному" методу [15], который представлен в разделе 4.1.

В последующих разделах задача $1 \mid r_j \mid \sum w_j U_j$ формулируется как задача *целочисленного линейного программирования* (ЦЛП) и решается *методом отсечения и ветвления* (branch and cuts).

Пусть булева переменная x_j принимает значение 1, если требование $j \in N$ обслуживается вовремя и 0, – если требование j запаздывает.

$$\begin{cases} \sum\limits_{j \in N} w_j(1 - x_j) \longrightarrow \min \\ \mathcal{R}_{D(x)}, \\ \texttt{disjunctive}(x), \\ x \in \{0,1\}^n. \end{cases}$$

Ограничение $\texttt{disjunctive}(x)$ выполняется тогда и только тогда, когда множество требований $J = \{j : x_j = 1\}$ может быть обслужено на одном

приборе без прерываний и с соблюдением времён поступления и директивных сроков, $\mathcal{R}_{D(x)}$ — релаксация ограничения `disjunctive`. В разделе 4.4 рассматриваются различные варианты релаксации.

Лемма 0.1 *Пусть для двух требований $i, j \in N$ выполняется $r_i < d_j$. Если вектор x удовлетворяет ограничению `disjunctive`, тогда x также удовлетворяет ограничению*[2]

$$\sum_{l \in N} \min[d_j - r_i, (p_l - \max\{\alpha_{li}, \beta_{jl}\})_+]x_l \leq d_j - r_i,$$

где $\alpha_{li} = (r_i - r_l)_+$ и $\beta_{jl} = (d_l - d_j)_+$.

Раздел 4.5 посвящён вопросу проверки на допустимость решения $\bar{x}$ задачи ЦЛП, а также вопросу построения отсечений в случае недопустимости $\bar{x}$. Алгоритм выполняется для множества требований $\bar{J} = \{j : \bar{x}_j = 1\}$ и в случае его недопустимости находится недопустимое подмножество $S \subseteq \bar{J}$, для которого строится ограничение

$$\sum_{j \in S} x_j \leq \mid S \mid -1.$$

Следующее утверждение представляет отсечения второго вида.

Лемма 0.2 *Пусть заданы такие множество требований $\Omega \subset N$ и требование $k \in N \setminus \Omega$, что k может быть обслужено только с момента времени r_k^Ω, $r_k^\Omega > r_k$, если все требования из Ω выполняются вовремя. Положим также $\alpha_{lk}^\Omega = (r_k^\Omega - r_l)_+$. Тогда вектор x, удовлетворяющий ограничению `disjunctive`, удовлетворяет неравенству*

$$\sum_{l \in N \setminus \Omega \setminus \{k\}} \min\left[d_j - r_k^\Omega, (p_l - \max\{\alpha_{lk}^\Omega, \beta_{jl}\})_+\right] x_l +$$
$$(p_k - \beta_{jk})_+ x_k \leq d_j - r_k^\Omega + (r_k^\Omega - r_k)\left(\mid \Omega \mid - \sum_{o \in \Omega} x_o\right),$$

для каждого требования $j \in N \setminus \Omega$, $d_j > r_k^\Omega$.

В работе предлагается алгоритм сложности $O(n^3)$ операций, который проверяет существование отсечения найденного вида. В разделе 4.6 рассматриваются различные варианты предложенного алгоритма отсечения и ветвления. В заключительном разделе 4.7 представлены результаты экспериментального исследования предложенного алгоритма на множестве общедоступных тестовых примеров. Показано, что разработанный алгоритм для задачи

[2] $(x)_+ = \left\{ {x, x>0, \atop 0, x \leq 0;} \right.$; $(x)_- = \left\{ {-x, x<0, \atop 0, x \geq 0;} \right.$; $|x| = (x)_+ + (x)_-$.

$1 \mid r_j \mid \sum w_j U_j$ является более эффективным, чем лучший алгоритм для данной задачи, существующий в литературе [180].

В **пятой главе** рассматриваются комбинаторные свойства NP-трудной в обычном смысле [114] задачи минимизации суммарного запаздывания для одного прибора $1 \mid\mid \sum T_j$. Получены свойства оптимальных расписаний, на основе которых построены алгоритмы решения задачи. Построены оценки оптимального значения целевой функции. В разделе 5.1 приводится постановка рассматриваемой задачи, вводятся необходимые понятия и определения.

В разделе 5.2 приводятся алгоритмы решения рассматриваемой задачи, основанные на "декомпозиционных" свойствах задачи. Одно из первых "декомпозиционных" свойств задачи было получено Лоулером, который предложил алгоритм решения задачи трудоёмкости $O(n^4 \sum p_j)$ операций [150].

Перенумеруем требования множества N в порядке неубывания директивных сроков $d_1 \leq \ldots \leq d_n$ (при этом, если $d_j = d_{j+1}$, то $p_j \leq p_{j+1}$, $j = 1, 2, \ldots, n-1$). Пусть j^* будет требованием с максимальной продолжительностью (если таких требований несколько, то выберем требование с наибольшим номером), т.е. $j^* = \max\{j \in N : p_j = \max\limits_{i \in N} p_i\}$. Пусть $S_k = t_0 + \sum\limits_{j=1}^{k} p_j, k = 1, 2, \ldots, n$, и $S = S_n$.

Через $L(I)$ обозначим множество индексов $k \geq j^*$, для которых выполнены неравенства $d_j + p_j < S_k, j = j^* + 1, \ldots, k$ и $S_k \leq d_{k+1}$, доопределив $d_{n+1} := +\infty$. Множество $L(I)$ будем называть *множеством подходящих позиций* для требования j^*.

Теорема 0.10 *[26] Для любого примера I множество подходящих позиций $L(I)$ для требования j^* не пусто и существует оптимальное расписание $\pi^* \in \Pi^*(I)$, при котором требование j^* обслуживается на позиции $k \in L(I)$. При этом для подходящей позиции k выполняется*

$$(j \to j^*)_{\pi^*} \text{ для } j \in \{1, 2, \ldots, k\} \setminus \{j^*\} \text{ и } (j^* \to j)_{\pi^*} \text{ для } j \in \{k+1, \ldots, n\}.$$

Идея алгоритма решения произвольного примера задачи с помощью "декомпозиционных" свойств (алгоритм A) основана на обходе дерева подпримеров "в глубину" с использованием рекурсивного вызова процедуры декомпозиции примеров на подпримеры. Если в двух различных вершинах дерева подпримеров будет находиться один и тот же подпример (одинаковы множества требований и моменты начала обслуживания), то алгоритм A построит оптимальное расписание для данного подпримера дважды. Чтобы избежать

этого недостатка, предлагается использовать алгоритм SBA, процесс выполнения которого разбивается на два этапа. На первом этапе осуществляется "разбиение" исходного примера вплоть до получения подпримеров из одного требования. Список подпримеров организован как хэш-таблица. Значение хэш-функции, соответствующее подпримеру, вычисляется с использованием момента начала обслуживания требований данного примера. На втором этапе осуществляется "сборка" оптимального расписания на основе построенного списка подпримеров. В заключение раздела приводятся результаты экспериментального сравнения работы алгоритмов A и SBA.

В разделе 5.3 рассматриваются свойства оптимальных расписаний и предлагается алгоритм решения примеров I в случае, когда при любом расписании $\pi \in \Pi(I)$ запаздывает одно и то же количество требований $m, 0 \leq m \leq n$. При этом запаздывающие требования упорядочены в конце расписания π (т.е. расписание π можно представить в виде $\pi = (\pi_1, \pi_2)$, где подрасписание π_2 содержит все m запаздывающих требований). Через $D(\pi)$ обозначим множество запаздывающих требований при расписании π. В рассматриваемом случае верны следующие утверждения.

Лемма 0.3 *При любом оптимальном расписании π^* требования множества $D(\pi^*)$ обслуживаются в SPT порядке*[3].

Далее, перенумеруем требования примера I в порядке невозрастания продолжительностей обслуживания $p_1 \geq p_2 \geq \ldots \geq p_n$, при этом, если $p_j = p_{j+1}, j = 1, \ldots, n-1$, то $d_j \geq d_{j+1}$.

Лемма 0.4 *Существует оптимальное расписание π^* такое, что если для некоторого $k \in N$ выполняется $k \in D(\pi^*)$, то $(j \to k)_{\pi^*}$ для всех требований $j \in \{k+1, \ldots, n\}$.*

Найденные свойства позволяют построить алгоритм нахождения оптимального расписания за $O(n^3)$ операций (алгоритм FA). Построение оптимального расписания алгоритмом FA сводится к обходу дерева, каждая ветка которого соответствует некоторому порядку обслуживания запаздывающих требований.

В разделе 5.4 приводятся две оценки оптимального значения суммарного запаздывания, основанные на свойствах SPT-расписаний. Полученные

[3]SPT – Shortest Processing Time. Расписание π называется SPT-расписанием, если требования упорядочены при расписании π в порядке неубывания продолжительностей обслуживания.

оценки позволяют найти абсолютную погрешность значения суммарного запаздывания при SPT-расписании[4].

Перенумеруем требования множества N в порядке неубывания продолжительностей обслуживания, т.е. $p_1 \leq p_2 \leq \ldots \leq p_n$. SPT-расписание $(1, 2, \ldots, n)$ будем обозначать через π_{spt}. Рассмотрим пример задачи $I = \langle \{p_j, d_j\}_{j \in N}, t_0 \rangle$ с оптимальным значением целевой функции $F^*(I)$. Пусть $d_{\min} = \min\limits_{j \in N} d_j$ и $d_{\max} = \max\limits_{j \in N} d_j$. Выберем величины C и δ такими, что

$$d_{\min} \leq C \leq d_{\max} \text{ и } \delta = \max\{d_{\max} - C, C - d_{\min}\}.$$

Рассмотрим пример $I' = \langle \{p_j, d'_j = C\}_{j \in N}, t_0 \rangle$.

Теорема 0.11 (*первая оценка*). *Для выбранных значений C и δ верно неравенство*

$$|F^*(I) - F^*(I')| \leq n\delta \left(2 - 2\frac{C}{S} + \frac{\delta}{S}\right) + \delta.$$

Для формулировки второй оценки рассмотрим функцию

$$f(t) = \sum_{j=1}^{n} \max \left\{ 0, t_0 + \sum_{i=1}^{j} p_i - t \right\}.$$

Значение этой функции в точке t равно оптимальному значению суммарного запаздывания для параметрического примера $I(t) = \langle \{p_j, t\}_{j \in N}, t_0 \rangle$, при котором директивные сроки всех требований равны t. Функция f является кусочно-линейной, невозрастающей, непрерывной и неотрицательной функцией, принимающей значения 0 для всех $t \geq t_0 + \sum\limits_{j \in N} p_j$.

Теорема 0.12 (*вторая оценка*). *Для любого примера I справедливы неравенства $f(d_{\max}) \leq F^*(I) \leq f(d_{\min})$.*

Раздел 5.5 посвящён результатам экспериментальных исследований задачи.

В работе применяется новая методика проведения экспериментов. В ϵ-окрестности произвольно выбранной начальной точки (примера) x_1 находим точку с большей трудоёмкостью (количеством ветвлений в дереве поиска) x_2. Затем в окрестности точки x_2 ищем "более сложную" точку и т.д. Процесс

[4]Относительная погрешность значения суммарного запаздывания при EDD-расписании (расписании, требования при котором упорядочены в порядке неубывания значений директивных сроков) установлена Лоулером в 1977 г.

останавливается, когда не удаётся найти "более сложную" точку, в окрестности текущей точки. Замечен интересный факт: для всех построенных алгоритмов "предельные сложные" точки ортогональны в $(2n+1)$-мерном пространстве...

В **шестой главе** рассматривается частный NP-трудный случай [6] задачи минимизации суммарного запаздывания

$$\begin{cases} p_1 \geq p_2 \geq \ldots \geq p_n, \\ d_1 \leq d_2 \leq \ldots \leq d_n. \end{cases} \qquad (6.1)$$

Предлагается подход решения задачи для этого случая, при котором исходное множество требований N разбивается на $\Bbbk$ таких непересекающихся подмножеств $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_\Bbbk$, что $N = \mathcal{M}_1 \bigcup \mathcal{M}_2 \bigcup \ldots \bigcup \mathcal{M}_\Bbbk$ и $\max\limits_{i,j \in \mathcal{M}_\alpha} |d_i - d_j| \leq \min\limits_{j \in \mathcal{M}_\alpha} p_j, \alpha = 1, \ldots, \Bbbk$, а затем на основе этого разбиения строится оптимальное расписание.

В разделе 6.1 формулируется и доказываются две леммы, отражающие свойства оптимальных расписаний исследуемого случая.

Лемма 0.5 *Существует оптимальное расписание π^*, при котором выполняется либо $(k \to i)_{\pi^*}$, либо $(j \to k)_{\pi^*}$ для любой тройки требований $i, j, k \in N$, что $|d_i - d_j| \leq \min\{p_i, p_j\}$, $k < \min\{i, j\}$ и $(i \to j)_{\pi^*}$.*

Лемма 0.6 *Существует оптимальное расписание π^*, при котором выполняется либо $(k \to j)_{\pi^*}$, либо $((k+1) \to k)_{\pi^*}$ для любой пары требований $k, j \in N$, $k < j$.*

На первом этапе решения задачи производится разбиение множества требований N на непересекающиеся подмножества $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_\Bbbk$, такие, что разность между максимальным и минимальным директивными сроками требований каждого подмножества не превышает величины минимальной продолжительности требований из этого подмножества[5].

Определим *параметрический пример* $I_k(t) = \langle \{p_j, d_j(t)\}_{j \in N_k}, 0 \rangle$, где $N_k = \{k, k+1, \ldots, n\}$ и $d_j(t) = d_j - d_n + t - t_0$. $I_k(t)$ – это пример исследуемой задачи обслуживания требований множества N_k с моментом начала обслуживания равным нулю и директивными сроками $d_j(t)$, линейно зависящими от параметра t.

[5]Такое разбиение исходного множества требований может быть выполнено за $O(n)$ операций.

Идея предлагаемого подхода заключается в построении оптимальных расписаний $\pi_k^*(t)$ для примеров $I_k(t)$, $k = n, n-1, \ldots, 1$, для всех целочисленных точек $t \in [t_0, d_n]$. Расписания $\pi_k^*(t)$ строятся на основе построенных на предыдущих шагах расписаний $\pi_l^*(t')$, $l > k$, $t' \in [t_0, d_n]$.

Согласно леммам, при построении оптимального расписания для исходного примера I можно исключить из рассмотрения все расписания π, при которых:

(a) $(i \to k \to j)_\pi$, $k < \min\{i, j\}$, $i, j \in M_\alpha, k \in M_\beta, \beta < \alpha$;

(b) $(j \to k \to (k+1))_\pi$, $k < j$.

В разделе 6.3 рассматривается случай $\Bbbk = 1$, когда параметры требований удовлетворяют условиям:

$$\begin{cases} p_1 \geq p_2 \geq \ldots \geq p_n; \\ d_1 \leq d_2 \leq \ldots \leq d_n; \\ d_n - d_1 \leq p_n. \end{cases}$$

В этом случае при построении оптимального расписания $\pi_k^*(t)$ с помощью описанного выше подхода для каждой точки t требуется просмотреть только две позиции для требования k: до всех требований множества N_{k+1}, где $N_{k+1} = \{k+1, \ldots, n\}$ и после всех требований этого множества. При обслуживании требования k до всех требований множества N_{k+1} получаем расписание $(k, \pi_{k+1}^*(t - p_k))$, после всех требований – расписание $(\pi_{k+1}^*(t), k)$. Расписание с наименьшим значением суммарного запаздывания среди двух полученных расписаний будет оптимальным расписанием для примера $I_k(t)$. Алгоритм, реализующий данный подход к построению расписания в этом случае, мы назвали *алгоритмом* B-1.

Теорема 0.13 *Алгоритм* B-1 *находит оптимальное расписание для данного* NP*-трудного случая задачи* $1||\sum T_j$ *за* $O(n\sum p_j)$ *операций.*

Напомним, здесь мы рассматриваем примеры с целыми, положительными временами обслуживания требований.

В разделе 6.4 рассматривается случай для произвольного значения $\Bbbk$. Приводится описание алгоритма B-$\Bbbk$ нахождения оптимального расписания в случае (6.1), трудоёмкость алгоритма $O(\Bbbk n \sum p_j)$ операций.

В разделе 6.5 рассматриваются примеры, параметры требований которых удовлетворяют условиям:

$$\begin{cases} d_1 \le d_2 \le \ldots \le d_n; \\ d_n - d_1 \le 1; \\ t_0 \in \mathbb{Z}, \end{cases}$$

предполагается, что $p_j \in \mathbb{Z}^+$, $j = 1, \ldots, n$.

Алгоритм C-1 находит оптимальное расписание для данного случая за $O(n^2)$ операций. Отметим, что алгоритм можно использовать для решения примеров в случае, когда в условиях второе неравенство $(d_n - d_1 \le 1)$ заменено на $d_n - d_1 \le \mathbf{НОД}\,(p_1, \ldots, p_n)$, – наименьший общий делитель среди всех продолжительностей обслуживания требований.

В разделе 6.6 рассматриваются примеры, параметры требований которых удовлетворяют условиям

$$d_j - d_{j-1} > p_j,\ j = 2, 3, \ldots, n.$$

Поскольку $p_j > 0$, $j \in N$, то выполняется $d_1 < \ldots < d_n$. Отметим, что этот случай является "предельным" подслучаем, когда количество подмножеств $\Bbbk = n$, т.е. каждое подмножество $\mathcal{M}_\alpha, \alpha = 1, \ldots, \Bbbk$, состоит ровно из одного требования, $\mu_\alpha = \{\alpha\}$.

Показано, что для данного случая задачи существует оптимальное расписание $\pi^* \in \Pi^*(I)$, при котором требование j^* обслуживается на первой позиции из множества $L(I)$. Данное свойство позволяет построить алгоритм B-n решения задачи трудоёмкости $O(n^2)$ операций.

В разделе 6.7 построен алгоритм трудоёмкости $O(n^3)$ операций для случая:

$$\begin{cases} p_1 \le p_2 \le \ldots \le p_n; \\ p_1 + d_1 < p_2 + d_2 < \ldots < p_2 + d_2. \end{cases}$$

В **седьмой главе** исследуются свойства алгоритма B-1. Показано, что данный алгоритм может быть использован для нахождения решения следующих NP-полных задач разбиения: РАЗБИЕНИЕ, ЧЁТНО–НЕЧЁТНОЕ РАЗБИЕНИЕ (ЧНР), ЧЁТНО–НЕЧЁТНОЕ РАЗБИЕНИЕ С ОГРАНИЧЕНИЯМИ (ОЧНР).

Расписание π назовем расписанием, обладающим *свойством* B-1, если для всех требований $k \in \{1, 2, \ldots, n-1\}$ при расписании π выполняется либо $(k \to j)_\pi$, либо $(j \to k)_\pi$ для всех $j \in \{k+1, \ldots, n\}$.

Множество всех расписаний, обладающих *свойством* B-1 для примера I, будем обозначать через $\Pi_{B-1}(I)$. Пусть $\Pi^*_{B-1}(I) = \Pi_{B-1}(I) \bigcap \Pi^*(I)$.

Теорема 0.14 *Алгоритм* B-1 *находит оптимальное расписание примера* I *задачи* $1 \,||\, \sum T_j$ *тогда и только тогда, когда* $\Pi^*_{B-1}(I) \neq \emptyset$.

Раздел 7.2 посвящён классической NP-полной задаче РАЗБИЕНИЕ, рассматривается схема полиномиального сведения примеров этой задачи к примерам задачи теории расписаний $1 \,||\, \sum T_j$.

В классической задаче РАЗБИЕНИЕ требуется определить, существует ли такое разбиение множества n целых положительных чисел на два подмножества, так что суммы чисел в обоих подмножествах равны. Задача ЧНР представляет собой модификацию задачи РАЗБИЕНИЕ, когда на разбиение множества накладывается ограничение, что соседние числа (первое и второе, третье и четвертое и т.д.) должны оказаться в разных подмножествах. Задача ОЧНР представляет собой задачу ЧНР с дополнительными ограничениями на значения чисел исходного множества.

Показано, что любое (в том числе и оптимальное) расписание канонического вида обладает свойством B-1. Таким образом, мы можем использовать алгоритм B-1 для нахождения решения канонических примеров задачи $1 \,||\, \sum T_j$ и, соответственно, задач РАЗБИЕНИЕ, ЧНР, ОЧНР. С учётом особенностей канонических примеров предложен алгоритм B-1-*канонический*, трудоёмкость которого не хуже известного алгоритма из книги Гэри и Джонсона [7] для задачи РАЗБИЕНИЕ.

В разделе 7.3 приводится модификация алгоритма B-1, которая позволяет снизить трудоёмкость решения примеров задачи $1 \,||\, \sum T_j$. Идея алгоритма B-1–*модифицированный* заключается в том, что процесс построения оптимального расписания сведён к процессу построения кусочно-линейной функции специального вида. Рассматриваются три операции над функциями: “сдвиг” функции, сложение двух функций и нахождение функции минимума двух функций. Полученные свойства рассматриваемых функций были использованы для построения алгоритма B-1–*модифицированный*. Преимущество этого алгоритма заключается в том, что при построении решения нет необходимости рассматривать все целочисленные точки интервала $t \in [t_0, d_n]$. Трудоёмкость решения полиномиально зависит от максимального количества точек изменения наклона функций, рассматриваемых в ходе решения. Преимуществом Алгоритма B-1–*модифицированный* является то, что при масштабировании с "небольшим" изменением всех параметров примера, трудоёмкость алгоритма решения не меняется. Также данный алгоритм может быть использован для решения задач $1 \,||\, \sum T_j$, РАЗБИЕНИЕ, ЧНР и ОЧНР с нецелочисленными параметрами.

В **восьмой главе** рассматривается графическая реализация метода динамического программирования на примерах решения задач РАЗБИЕНИЕ и РАНЕЦ. Проведён сравнительный анализ предлагаемого метода с известными алгоритмами решения этих задач.

Задача РАЗБИЕНИЕ. Задано упорядоченное множество из n положительных целых чисел $B = \{b_1, b_2, \ldots, b_n\}$, $b_1 \geq b_2 \geq \ldots \geq b_n$. Требуется разбить множество B, на два подмножества B_1 и B_2, $B_1 \bigcup B_2 = B, B_1 \bigcap B_2 = \emptyset$, так, чтобы минимизировать значение:

$$|\sum_{b_i \in B_1} b_i - \sum_{b_i \in B_2} b_i| \longrightarrow \min .$$

Задача об одномерном РАНЦЕ (0-1 Knapsack).

$$\begin{cases} f(x) = \sum\limits_{i=1}^{n} p_i x_i \longrightarrow \max \\ \sum\limits_{i=1}^{n} w_i x_i \leq C, \\ x_i \in \{0,1\},\ i = 1, \ldots, n. \end{cases}$$

Когда $p_i = w_i = b_i, i = 1, 2, \ldots, n$, и $C = \frac{1}{2} \sum\limits_{j=1}^{n} b_j$, то данные задачи являются эквивалентными.

В разделе 8.1 представлен графический алгоритм решения задачи РАЗБИЕНИЕ. В алгоритме последовательно на шаге $\alpha = 1, 2, \ldots, n$ рассматриваются следующие функции:

$$F_\alpha^1(t) = |\sum_{b_j \in B_1(t-b_\alpha)} b_j - \sum_{b_j \in B_2(t-b_\alpha)} b_j|;$$

$$F_\alpha^2(t) = |\sum_{b_j \in B_1(t+b_\alpha)} b_j - \sum_{b_j \in B_2(t+b_\alpha)} b_j|.$$

На каждом шаге α алгоритма $B_1(t) \bigcup B_2(t) = \{b_1, b_2, \ldots, b_{\alpha-1}\}$, для каждого t. Последовательно "добавляется" очередное число b_α, где $\alpha = 1, 2, \ldots, n$, в множество $B_1(t)$ или $B_2(t)$. В каждой точке t "добавление" числа b_α выбирается таким образом, чтобы минимизировать значение функции $|\sum\limits_{b_j \in B_1(t)} b_j + t - \sum\limits_{b_j \in B_2(t)} b_j|$. На очередном шаге α из функции $F_{\alpha-1}(t) = |\sum\limits_{b_j \in B_1(t)} b_j - \sum\limits_{b_j \in B_2(t)} b_j|$, полученной на предыдущем $\alpha - 1$ шаге, строится функция

$$F_\alpha(t) = \min\{F_{\alpha-1}(t - b_\alpha), F_{\alpha-1}(t + b_\alpha)\} = \min\{F_\alpha^1(t), F_\alpha^2(t)\},$$

где $F_0(t) = 0,\ \forall\ t$. Если $F_\alpha^1(t) < F_\alpha^2(t)$, то $B_1(t) = B_1(t - b_\alpha) \bigcup \{b_\alpha\}$, иначе $B_2(t) = B_2(t + b_\alpha) \bigcup \{b_\alpha\}$. Кусочно-линейную функцию $F_\alpha(t)$ можно представить (и хранить) в табличном виде по точкам "излома": $t_0; t_1; \ldots; t_{m_\alpha}$. На промежутке $[t_i - \frac{t_i - t_{i-1}}{2}, t_i + \frac{t_{i+1} - t_i}{2}), i = 1, 2, \ldots, m_\alpha - 1$, кусочно-линейная функция $F_\alpha(t)$ задаётся уравнением $F_\alpha(t) = |t - t_i|$, т.е. график функции пересекает ось t в точке t_i. Каждому временно́му интервалу $[t_i - \frac{t_i - t_{i-1}}{2}, t_i + \frac{t_{i+1} - t_i}{2})$ соответствует некоторое фиксированное разбиение $(\overline{B}_1; \overline{B}_2)$, т.е. $B_1(t^1) = B_1(t^2) = \overline{B}_1,\ \forall t^1, t^2 \in [t_i - \frac{t_i - t_{i-1}}{2}, t_i + \frac{t_{i+1} - t_i}{2})$ (аналогично и для $B_2(t)$). Пусть на предыдущем шаге $\alpha - 1$ получена функция $F_{\alpha-1}(t)$, заданная таблично. На шаге α мы рассматриваем две функции $F_{\alpha-1}(t - b_\alpha)$ и $F_{\alpha-1}(t + b_\alpha)$, заданные двумя таблицами в точках "излома": $t_0 - b_\alpha$, $t_1 - b_\alpha$, $\ldots t_i - b_\alpha$, $\ldots$, $t_{m_{\alpha-1}} - b_\alpha$ и $t_0 + b_\alpha$, $t_1 + b_\alpha$, $\ldots$, $t_i + b_\alpha$, $\ldots$, $t_{m_{\alpha-1}} + b_\alpha$.

В разделе 8.2 показано как можно производить сокращение рассматриваемых интервалов. Учитывая, что на шаге n нам требуется вычислить значение целевой функции и найти разбиение лишь в точке $t = 0$, то на шаге $n - 1$ нам достаточно рассчитать значения в точках $t \in [-b_n, b_n]$. Аналогично на шаге $n - 2$ достаточно рассмотреть интервал $[-b_n - b_{n-1}, b_n + b_{n-1}]$ и т.д.

Следовательно, на каждом шаге α достаточно рассматривать интервал $[-\sum\limits_{j=\alpha+1}^{n} b_j,\ \sum\limits_{j=\alpha+1}^{n} b_j]$, вместо интервала $[-\sum\limits_{j=1}^{n} b_j,\ \sum\limits_{j=1}^{n} b_j]$.

При построении функции $F_\alpha(t)$ рассматриваются только точки интервала $t \in [-\sum\limits_{j=\alpha+1}^{n} b_j,\ \sum\limits_{j=\alpha+1}^{n} b_j]$. Чтобы интервал сокращался максимально быстро необходимо упорядочить b_j по невозрастанию. Учитывая, что функция $F_\alpha(t), \alpha = 1, 2, \ldots, n$, является чётной, поэтому достаточно хранить "половину" таблицы. Кроме того, следует отметить, что при "масштабировании с небольшим изменением параметров" примера, т.е. когда $b'_j = Kb_j + \varepsilon_j$, где $|\varepsilon_j| \ll K$, $K-$ некоторая достаточно большая положительная константа, $j = 1, 2, \ldots, n$, трудоёмкость алгоритма, построенном на методе динамического программирования [1], составит $O(Kn \sum b_j)$ операций, в то время как у графического алгоритма трудоёмкость не изменится. Для алгоритма *Balsub* [137] с трудоёмкостью $O(nb_{\max})$ операций такое "масштабирование" примера также приведёт к увеличению трудоёмкости в K раз. Таким образом, графический алгоритм находит решение за одно и тоже количество операций для всех точек некоторого конуса в n-мерном пространстве, если представить параметры примера как точку $(b_1, b_2, \ldots, b_n)$ в n-мерном пространстве. Причём параметры примера могут быть как отрицательными, так и нецелочисленными.

Существуют классы примеров, для которых трудоёмкость графического алгоритма растёт экспоненциально быстро (от n):

1. $B = \{b_1, b_2, \ldots, b_n\} = \{M, M-1, M-2, \ldots, 1, 1, \ldots, 1\}$, где M – достаточно большое число, сумма единиц в примере равна $M(M+1)/2$, т.е. $n = M + M(M+1)/2$;

2. Класс нецелочисленных примеров $B = \{b_1, b_2, \ldots, b_n\}$, если не существует такого набора чисел $\lambda_i = \pm 1, i = 1, \ldots, n$, что выполняется $\lambda_1 b_1 + \lambda_2 b_2 + \ldots + \lambda_n b_n = 0$.

В разделе 8.3 представлена графическая реализация метода динамического программирования решения задачи РАНЕЦ. Также как и для задачи РАЗБИЕНИЕ в ранец последовательно "добавляются предметы". На шаге $(\alpha+1)$ строится график $g^2(t)$ из графика $g_\alpha(t)$ смещением "наверх" на $p_{\alpha+1}$ и "вправо" на $w_{\alpha+1}$. График $g^1(t)$ полностью повторяет график $g_\alpha(t)$. Следовательно, чтобы построить $g_{\alpha+1}(t) = \max\{g^1(t), g^2(t)\}$ необходимо рассмотреть не более $2m_\alpha$ интервалов, образованных точками из множества $\{t_0, t_1, \ldots, t_{m_\alpha}, t_0 + a_{\alpha+1}, t_1 + a_{\alpha+1}, \ldots, t_{m_\alpha} + a_{\alpha+1}\}$, которые принадлежат интервалу $[0, C]$. Количество точек не превосходит C для $w_i \in \mathbb{Z}^+$, $i = 1, \ldots, n$. Параметры задачи могут быть и отрицательными.

Идея графического алгоритма была успешно использована и для решения задачи многомерный РАНЕЦ ($d-m$-dimensional Knapsack):

$$\begin{cases} f(x) = \sum\limits_{i=1}^{n} p_i x_i \longrightarrow \max \\ \sum\limits_{i=1}^{n} w_{ij} x_i \le C_j, j = 1, \ldots, d_i, \\ x_i \in \{0, 1, \ldots, m_i\},\ i = 1, \ldots, n. \end{cases}$$

В **девятой главе** рассматриваются задачи минимизации обобщенной функции запаздывания.

Обобщенное запаздывание задается следующим образом:

$$GT_j(\pi) = \begin{cases} 0, & \text{если } C_j(\pi) - d_j \le 0, \\ v_j \cdot (C_j(\pi) - d_j), & \text{если } 0 < C_j(\pi) - d_j \le b_j, \\ w_j, & \text{если } b_j < C_j(\pi) - d_j, \end{cases}$$

где $w_j \ge v_j b_j$ для каждого $j \in N$. Целевая функция задачи:

$$F(\pi) = \sum_{j=1}^{n} GT_j(\pi) \to \min .$$

Показано, что даже в случае

$$b_j = p_j, \quad v_j = 1, \quad w_j = p_j, \tag{1}$$

т.е., $GT_j(\pi) = \min\{\max\{0, C_j(\pi) - d_j\}, p_j\}$ для всех $j \in N$ задача является NP-трудной.

Теорема 0.15 *Частный случай (1) задачи* $1\|\sum GT_j$ *является NP-трудным.*

Для поиска оптимального решения можно ограничиться следующей структурой расписания.

Лемма 0.7 *Для частного случая (1) существует оптимальное расписание вида* $\pi = (G, H)$*, где для всех требований* $i \in G$ *выполняется* $0 \le GT_i(\pi) < p_i$*, а для требований* $j \in H$ *выполняется* $GT_i(\pi) = p_i$*. Требования из множества* G *обслуживаются в порядке EDD*[6]*, а требования из множества* H *– в порядке LDD*[7]*.*

Также в главе 9 рассмотрены задачи, в которых строится расписание с максимальным значением целевой функции. Формулируются эти задачи следующим образом, практически совпадающим с формулировками классических задач минимизации.

Необходимо обслужить n требований на одном приборе. Прерывания при обслуживании и обслуживание более одного требования в любой момент времени запрещены. Для каждого требования $j \in N = \{1, 2, \ldots, n\}$ заданы продолжительность обслуживания $p_j > 0$ и директивный срок его окончания d_j, где N – множество требований, которые необходимо обслужить. Прибор начинает обслуживание требований с момента времени 0. Простои прибора при обслуживании требований запрещены (иначе задачи максимизации становятся тривиальными). Расписание обслуживания требований $\pi = (j_1, j_2, \ldots, j_n)$ строится с момента времени 0 и однозначно задаётся перестановкой элементов множества N. Обозначим через $C_{j_k}(\pi) = \sum_{l=1}^{k} p_{j_l}$ время завершения обслуживания требований j_k при расписании π. Если $C_j(\pi) > d_j$, тогда требование j запаздывает, и в этом случае полагают $U_j = 1$. Если $C_j(\pi) \le d_j$, тогда требование j не запаздывает, и $U_j = 0$.

[6]EDD—earliest due date
[7]LDD—latest due date

Таблица 1: Сведения о трудоёмкости задач максимизации

Задача	трудоёмкость и алгоритмы	трудоёмкость соотв. задачи минимизации
$1(nd)\|\|\max\sum U_j$	Полин. разрешима за время $O(n\log n)$	Полин. разрешима за время $O(n\log n)$
$1(nd)\|D_j\|\max\sum T_j$	NP-трудна	Неизвестна
$1(nd)\|D_j\|\max\sum U_j$	NP-трудна	Неизвестна
$1(nd)\|D_j\|\max\sum w_jC_j$	NP-трудна	Неизвестна
$1(nd)\|D_j\|\max\sum C_j$	NP-трудна	Неизвестна
$1(nd)\|\|\max\sum w_jU_j$	NP-трудна	NP-трудна
$1(nd)\|r_j\|\max\sum U_j$	NP-трудна	NP-трудна
$1(nd)\|r_j\|\max\sum w_jC_j$	NP-трудна	NP-трудна
$1(nd)\|r_j\|\max\sum C_j$	Полин. разрешима за время $O(n\log n)$	NP-трудна
$1(nd)\|prec,r_j\|\max C_{\max}$	$O(n^2)$	NP-трудна
$1(nd)\|prec,r_j\|\max f_{\max}$	$O(n^4)$	NP-трудна
$1(nd)\|r_j\|\max f_{\max}$	$O(n^3)$	NP-трудна
$1(nd)\|prec,r_j\|\max\sum C_j$	NP-трудна	NP-трудна
$1(nd)\|prec,r_j\|\max\sum U_j$	NP-трудна	NP-трудна
$1(nd)\|\|\max\sum w_jT_j$	NP-трудна, Алгоритм решения трудоёмкости $O(n\min\{\sum w_j, d_{\max}\})$	NP-трудна
$1(nd)\|r_j\|\max\sum w_jT_j$	NP-трудна	NP-трудна
$1(nd)\|r_j\|\max\sum T_j$	NP-трудна	NP-трудна
$1(nd)\|\|\max\sum T_j$	Алгоритм решения трудоёмкости $O(n\sum p_j)$. Алгоритм решения трудоёмкости $O(n^2)$	NP-трудна. Алгоритм решения трудоёмкости $O(n^4\sum p_j)$
$1(sa)\|r_j\|\max\sum T_j$	Полин.разрешима за время $O(n^3)$	Неизвестна
$1(sa)\|r_j\|\max\sum w_jT_j$	NP-трудна	Неизвестна

Требуется построить расписание π^* обслуживания требований множества N, при котором достигается максимум функции $F(\pi)=\sum\limits_{j=1}^{n}U_j(\pi)$. Обозначим данную задачу через $1(nd)||\max\sum U_j$. Аналогично, обозначается задача максимизации суммарного запаздывания $1(nd)||\max\sum T_j$ и другие задачи максимизации. Запись (nd) означает, что рассматриваются только расписания, при которых прибор не простаивает (*non-delay*), т.е. все требования обслуживаются в интервале $[0,\sum p_j]$ без перерывов.

Для решения задачи $1(nd)||\max\sum T_j$ предложен графический алгоритм полиноминальной трудоемкости.

Теорема 0.16 *Алгоритм GrA решения задачи* $1(nd)||\max\sum T_j$ *имеет тру-*

доёмкость $O(n^2)$ операций.

В главе также рассмотрены задачи с одним невозобновимым ресурсом. Постановка этих задач совпадает с классической постановкой одноприборных задач. Дополнительно заданы невозобновимый ресурс G (деньги, топливо и т.п.) и моменты времени поступления ресурса $\{t_0, t_1, \ldots, t_y\}$, $t_0 = 0$, $t_0 < t_1 < \ldots < t_y$. В каждый момент времени t_i, $i = 0, 1, \ldots, y$, поступает $G(t_i) \geq 0$ единиц ресурса. Для каждого требования $j \in N$, задано потребление ресурса $g_j \geq 0$, которое происходит в момент начала обслуживания. Выполняется равенство

$$\sum_{j=1}^{n} g_j = \sum_{i=0}^{y} G(t_i).$$

Обозначим через S_j время начала обслуживания требования j. Расписание $S = (S_{j_1}, S_{j_2}, \ldots, S_{j_n})$ описывает в том числе порядок обслуживания требований: $\pi = (j_1, j_2, \ldots, j_n)$. Расписание $S = (S_{j_1}, S_{j_2}, \ldots, S_{j_n})$ является допустимым, если выполняется, помимо классических условий, следующее неравенство:

$$\sum_{k=1}^{i} g_{j_k} \leq \sum_{\forall l:\ t_l \leq S_{j_i}} G(t_l), \quad i = 1, 2, \ldots, n.$$

Перестановку π также называют расписанием, т.к. по этой перестановке можно вычислить расписание $S = (S_{j_1}, S_{j_2}, \ldots, S_{j_n})$ за время $O(n)$ операций.

Такие одноприборные задачи обозначают $1|NR, ...|...$, где NR обозначает присутствие в условии задачи невозобновимого ресурса (non-renewable).

Теорема 0.17 *Задачи* $1|NR|C_{\max}$, $1|NR, d_j = d|\sum T_j$, $1|NR|\sum U_j$ *и* $1|NR|L_{\max}$ *являются NP-трудными в сильном смысле.*

В **Заключении** перечислены нерешенные задачи и намечены пути дальнейших исследований.

Предложенная читателю монография продолжает серию книг по теории расписаний:
- Танаев В.С., Шкурба В.В. Введение в теорию расписаний // М.: Наука. Гл. ред. Физ.-мат. лит., 1975.-256 с.;
- Танаев В.С., Гордон В.С., Шафранский Я.М. Теория расписаний. Одностадийные системы // М.: Наука. Гл. ред. Физ.-мат. лит., 1989.- 384 с.;
- Танаев В.С., Сотсков Ю.Н., Струсевич В.А. Теория расписаний. Многостадийные системы // М.: Наука. Гл. ред. Физ.-мат. лит., 1989.- 328 с.;

- ТанаевВ.С., Ковалев М.Я., Шафранский Я.М. Теория расписаний. Групповые технологии // Минск: Институт технической кибернетики НАН Беларуси, 1998.- 290 с.
Автором представлены результаты многолетних научных исследований фундаментальных задач теории расписаний с минимаксными и суммарными критериями.

Работа выполнена при финансовой поддержке грантом Российского Научного Фонда. Проект № 17—19—01665.

Благодарности

Автор считает своим долгом выразить благодарность В.С. Танаеву, С.Н. Васильеву, Ю.И. Журавлёву, В.К. Леонтьеву, И.Х. Сигалу, Ю.А. Флёрову, Л.С. Иртышёвой, К.В. Рудакову, В.Н. Буркову, Ф.Т. Алескерову, Д.А. Новикову, С.В. Севастьянову, М.Я. Ковалёву, Я.М. Шафранскому, Ю.Н. Сотскову, Н.Н. Кузюрину, В.В. Шкурбе, А.А. Сапоженко, М.Н. Вялому, С.П. Тарасову, Р.Р. Садыкову, А.Г. Кварацхелия, Р.Р. Сираеву, Е.Р. Гафарову, С.А. Скиндереву, А.Н. Черных, Я.И. Заботину, О.Н. Шульгиной, Р.Ф. Хабибуллину, А.А. Корбуту, П. Брукеру, Ф. Вернеру (Германия), Ф. Баптисте (Франция), В. Шварцу (США) за помощь в работе и обсуждение полученных результатов.
При подготовке материалов данной книги большую помощь оказали Т.С. Ефимова, И.П.Голубева, Н.А.Правдивец и Е.Г. Лазарева.

Глава 1

Оценка абсолютной погрешности задач минимизации максимального временно́го смещения

1.1 Постановка задачи $1 \mid r_j \mid L_{\max}$

Будем рассматривать фундаментальную следующую задачу теории расписаний. На одном приборе необходимо обслужить требования множества $N = \{1, 2, \ldots, n\}$. Каждое требование мы будем обозначать его номером, то есть запись "требование j" эквивалентна записи "требование с номером j". Запрещается одновременное обслуживание и прерывания при обслуживании требований. Для требования $j \in N$ имеем: r_j – минимально возможный момент начала обслуживания, $p_j \geq 0$ – продолжительность обслуживания, d_j – директивный срок завершения обслуживания[1].

Расписание задаётся совокупностью $\pi = \{s_j \mid j \in N\}$ моментов начала обслуживания требований, также через π, τ будем обозначать перестановку $(j_1, \ldots, j_n)$ элементов множества N. Множество всевозможных расписаний обслуживания требований множества N обозначим как $\Pi(N)$. Расписание π называется *допустимым*, если момент начала обслуживания $s_j(\pi) \geq r_j$, $\forall j \in N$. Будем обозначать момент завершения обслуживания требования $j \in N$ в расписании π как $C_j(\pi)$. В теории расписаний принята англоязычная терминология, поэтому s_j, C_j — от *start time* и *completion time*. Разность $L_j(\pi) = C_j(\pi) - d_j$, $j \in N$, обозначает временно́е смещение требования j в расписании π (*lateness*). Максимальное временно́е смещение требований мно-

[1]Термину "директивный срок" в англоязычной литературе по теории расписаний соответствуют "*due date*" и "*deadline*", когда "можно" и "нельзя" нарушать директивный срок. В данной работе директивный срок "можно" нарушать, т.е. "*due date*".

жества N при расписании π определяется как

$$L_{\max}(\pi) = \max_{j \in N}\{C_j(\pi) - d_j\}.$$

Обозначим момент завершения обслуживания всех требований множества N при расписании π через

$$C_{\max}(\pi) = \max_{j \in N} C_j(\pi).$$

В англоязычной литературе $C_{\max}$ называют *makespane.*

Задача состоит в нахождении оптимального расписания π^* с наименьшим значением максимального временно́го смещения:

$$L^*_{\max} = \min_{\pi \in \Pi(N)} L_{\max}(\pi) = L_{\max}(\pi^*). \tag{1.1}$$

Для произвольного множества требований $M \subseteq N$ будем также обозначать:

$$r_M = \min_{j \in M} r_j, \quad d_M = \max_{j \in M} d_j, \quad p_M = \sum_{j \in M} p_j.$$

В стандартной нотации Грэхэма и др. (Graham at al. [125]) данная задача обозначается как $1 \mid r_j \mid L_{\max}$. Интенсивные работы над решением этой задачи продолжаются с начала 50-х годов 20-го века. Ленстра и др. (Lenstra et al. [167]) показали, что общий случай задачи $1 \mid r_j \mid L_{\max}$ является NP-трудным в сильном смысле. С теорией сложности можно подробнее познакомиться в работах [7], [141], [107], [170], [177].

Был выделен ряд полиномиально разрешимых случаев задачи, начиная с раннего результата Джексона (Jackson [133]) для случая $r_j = \text{const}$, $j \in N$, когда решением является расписание, в котором требования упорядочены по неубыванию директивных сроков (по правилу EDD). Такое расписание также будет оптимальным для случая, когда времена поступления и директивные сроки согласованы ($r_i \leq r_j \Leftrightarrow d_i \leq d_j$, $\forall i, j \in N$).

Расписание построенное по расширенному правилу Джексона (расписание Шраге): на очередное место в расписание ставится требование с минимальным директивным сроком среди уже поступивших на обслуживание и ещё не упорядоченных; если таких требований нет, то выбирается требование с минимальным временем поступления среди неупорядоченных.

Поттс (Potts [186]) представил итерационную версию расширенного правила Джексона (IJ) [133] и доказал, что $\frac{L_{\max}(\pi_{IJ})}{L^*_{\max}} \leq \frac{3}{2}$. Холл и Шмойс (Hall, Shmoys [127]) модифицировали итерационную версию и создали алгоритм

(MIJ), который гарантирует оценку $\frac{L_{\max}(\pi_{MIJ})}{L^*_{\max}} \leq \frac{4}{3}$. Также они представили две аппроксимационные схемы, гарантирующие нахождение ε-приближённого решения за $O(n \log n + n(1/\varepsilon)^{O(1/\varepsilon^2)})$ и $O((n/\varepsilon)^{O(1/\varepsilon)})$ операций. Мастролилли (Mastrolilli [172]) представил улучшенную аппроксимационную схему, трудоёмкость $O(n + (1/\varepsilon)^{O(1/\varepsilon)})$ операций.

В 1978 году Симонс (Simons [202]) показала, что задача $1 \mid r_j; p_j = p \mid L_{\max}$ полиномиально разрешима.

Специальные случаи $1 \mid prec; r_j \mid C_{\max}$, $1 \mid prec; p_j = p; r_j \mid L_{\max}$ и $1 \mid prec; r_j; pmtn \mid L_{\max}$ с ограничениями предшествования для требований были рассмотрены в работах Лоулера (Lawler [144]), Симонса (Simons [202]), Бейкера и др. (Baker et al. [78]). Хогевен (Hoogeveen [130]) предложил полиномиальный алгоритм для специального случая, когда параметры требований удовлетворяют ограничениям $d_j - p_j - A \leq r_j \leq d_j - A$, $\forall i$, для некоторой константы A. Псевдополиномиальный алгоритм для NP-сложного случая, когда времена поступления и директивные сроки расположены в обратном порядке ($d_1 \leq \ldots \leq d_n$ и $r_1 \geq \ldots \geq r_n$), был разработан в [57].

1.2 Обозначения и определения основных понятий

В данном разделе мы приведём основные обозначения и определения, которые будут нами использоваться в дальнейшем.

Будем обозначать через $L_j^A(\pi)$ и $C_j^A(\pi)$ временно́е смещение и время окончания обслуживания требования $j \in N$ при расписании π для примера A с параметрами требований $\{r_j^A, p_j^A, d_j^A\}$, $j \in N$. Соответственно, $L_{\max}^A(\pi) = \max\limits_{j \in N} L_j^A(\pi)$ – максимальное временно́е смещение расписания π для примера A.

Определение 1.1 *Для примера A каждая перестановка τ требований множества N однозначно определяет раннее расписание π_τ^A. В раннем расписании каждое требование $j \in N$ начинает обслуживаться сразу после окончания обслуживания предыдущего требования в соответствующей перестановке. Если время окончания обслуживания предыдущего требования меньше времени поступления текущего требования, то начало обслуживания откладывается до момента поступления. То есть, если $\tau = (j_1, j_2, \ldots, j_n)$, тогда $\pi_\tau^A = (s_{j_1}, s_{j_2}, \ldots, s_{j_n})$, где*

$$s_{j_1} = r_{j_1}^A, \quad s_{j_k} = \max\left\{s_{j_{k-1}} + p_{j_{k-1}}^A, r_{j_k}^A\right\}, \ k = 2, \ldots, n.$$

Ранние расписания будут играть в наших построениях важную роль, поскольку оптимальное расписание любого примера достаточно искать во множестве только ранних расписаний.

Через τ^A и π^A обозначим оптимальную перестановку и оптимальное расписание примера A. В качестве оптимальных расписаний будут рассматриваться только ранние, т.е. $\pi^A = \pi^A_{\tau^A}$.

Будем обозначать через $\Pi(N)$ множество перестановок требований множества N, а через Π_A – множество допустимых расписаний примера A. Для задачи $1|r_j|L_{\max}$ множество $\Pi(N)$ содержит $n!$ перестановок. Так как $n!$ растет очень быстро с ростом размерности задачи n, то алгоритм, при котором перебираются все $n!$ перестановок, явно не приемлим при нахождении оптимальной перестановки при решении практических задач большой размерности.

Определение 1.2 *Пусть задан пример A на множестве требований N. Будем говорить, что пример B на том же множестве требований наследует у примера A параметр x, если $x^B_j = x^A_j,\ \forall\ j \in N$.*

Определение 1.3 *Пример $Q = \{(r^Q_j, p^Q_j, d^Q_j)\,|\,j \in N\}$ называется инверсным к примеру $P = \{(r^P_j, p^P_j, d^P_j)\,|\,j \in N\}$, если выполнены соотношения*

$$r^Q_j = -d^P_j,\ p^Q_j = p^P_j,\ d^Q_j = -r^P_j, \forall\ j \in N.$$

Перестановка $\tau' = (i_n, i_{n-1}, \ldots, i_1)$ называется инверсной к перестановке $\tau = (i_1, \ldots, i_n)$.
Расписание $\pi' = \{s'_j\,|\,j \in N\}$ называется инверсным к расписанию $\pi = \{s_j\,|\,j \in N\}$, если $\forall\ j \in N$, $s'_j = -s_j - p_j$.

Нетрудно заметить, что понятие инверсии является симметричным, т.е. если пример X является инверсным к Y, то и Y инверсен к X. В частности, интервалы I_j и I'_j обслуживания требования j во взаимно-инверсных расписаниях S и S' симметричны относительно нуля.

Определение 1.4 *Расписание π, допустимое для заданного примера $V = \{(r^V_j, p^V_j, d^V_j)\,|\ \ j \in N\}$, будем называть* вполне допустимым, *если каждое требование $j \in N$ обслуживается в своём директивном интервале $[r^V_j, d^V_j]$.*

Кроме того, через $\Delta = \Delta(V, \pi)$ будем обозначать минимальную величину (возможно, отрицательную), которую следует прибавить ко всем директивным срокам требований примера V, чтобы допустимое расписание π стало вполне допустимым для полученного примера $V(\Delta)$. Очевидно, $\Delta(V, \pi) = L^V_{\max}(\pi)$.

Определение 1.5 *Для двух произвольных примеров A и B определим следующие функции*[2]*:*

$$\rho_d(A,B) = \max_{j\in N}\left\{d_j^A - d_j^B\right\} + \max_{j\in N}\left\{d_j^B - d_j^A\right\};$$

$$\rho_r(A,B) = \max_{j\in N}\left\{r_j^A - r_j^B\right\} + \max_{j\in N}\left\{r_j^B - r_j^A\right\};$$

$$\rho(A,B) = \rho_d(A,B) + \rho_r(A,B).$$

1.3 Абсолютная погрешность приближённого решения

Пусть задан произвольный пример A задачи $1 \mid r_j \mid L_{\max}$, и пример C наследует у примера A продолжительности обслуживания. Пусть также π^C – оптимальное расписание для примера C. Для исходного примера теорема об абсолютной погрешности оценивает сверху разницу между значением целевой функции для расписания π^C и оптимальным значением. На основе данной теоремы в разделе 1.4 будет представлена полиномиальная схема нахождения приближённого решения рассматриваемой задачи. Результаты, полученные в данном и следующем разделах, были опубликованы в [24, 49, 162].

Лемма 1.1 *Пусть пример B наследует у примера A моменты поступления требований и их длительности обслуживания. Тогда для любого допустимого расписания π верно соотношение*

$$L_{\max}^B(\pi) - L_{\max}^A(\pi) \le \max_{j\in N}\left\{d_j^A - d_j^B\right\}. \tag{1.2}$$

Доказательство. Действительно, для любого $i \in N$ и любого расписания π имеем:

$$L_{\max}^A(\pi) + \max_{j\in N}\left\{d_j^A - d_j^B\right\} \ge C_i(\pi) - d_i^A + d_i^A - d_i^B = C_i(\pi) - d_i^B.$$

Следовательно,

$$L_{\max}^A(\pi) + \max_{j\in N}\left\{d_j^A - d_j^B\right\} \ge \max_{i\in N}\{C_i(\pi) - d_i^B\} = L_{\max}^B(\pi).$$

□

[2]В последующих разделах данной главы мы "добавим" функцию $\rho_p(A,B) = \sum\limits_{j\in N}|p_j^A - p_j^B|$ и определим $\rho(A,B) = \rho_d(A,B) + \rho_r(A,B) + \rho_p(A,B)$.

Лемма 1.2 *Пусть пример B наследует у примера A моменты поступления требований и их длительности обслуживания (т.е., $r_j^A = r_j^B$ и $p_j^A = p_j^B$, $\forall j \in N$) и пусть π^A и π^B – оптимальные расписания примеров A и B, соответственно, а $\tilde{\pi}^B$ – приближённое решение примера B, удовлетворяющее условию*[3]

$$L^B_{\max}(\tilde{\pi}^B) - L^B_{\max}(\pi^B) \le \delta_B. \tag{1.3}$$

Тогда

$$0 \le L^A_{\max}(\tilde{\pi}^B) - L^A_{\max}(\pi^A) \le \rho_d(A,B) + \delta_B,$$

где $\rho_d(A,B) = \max\limits_{j\in N}\{d_j^A - d_j^B\} + \max\limits_{j\in N}\{d_j^B - d_j^A\}$.

Доказательство.

$$\begin{aligned}
& 0 \le L^A_{\max}(\tilde{\pi}^B) - L^A_{\max}(\pi^A) \overset{(1.2)}{\le} \\
\overset{(1.2)}{\le}\ & L^B_{\max}(\tilde{\pi}^B) - L^A_{\max}(\pi^A) + \max_{j\in N}(d_j^B - d_j^A) \overset{(1.2),(1.3)}{\le} \\
\overset{(1.2),(1.3)}{\le}\ & \delta_B + L^B_{\max}(\pi^B) + \max_{j\in N}(d_j^B - d_j^A) - L^B_{\max}(\pi^A) + \max_{j\in N}(d_j^A - d_j^B) \le \\
\le\ & \delta_B + \rho_d(A,B),
\end{aligned}$$

т.к. $L^B_{\max}(\pi^B) \le L^B_{\max}(\pi)$ для любого расписания π. Лемма доказана. □

Поскольку $\rho_d(A,B) = \rho_d(B,A)$, поменяв местами A и B, из леммы 1.2 получаем

Следствие 1.1 $0 \le L^B_{\max}(\pi^A) - L^B_{\max}(\pi^B) \le \rho_d(A,B)$. □

Лемма 1.3 *Пусть V и W — взаимно инверсные примеры со множеством требований N, а τ и τ' — взаимно инверсные перестановки из $\Pi(N)$. Тогда $L^V_{\max}(\pi_\tau) = L^W_{\max}(\pi_{\tau'})$.*

Доказательство. Пусть $\Delta = \Delta(V, \pi_\tau)$. Поскольку расписание π_τ вполне допустимо для примера $V(\Delta)$, то расписание π', инверсное к π_τ, вполне допустимо для примера W', инверсного к примеру $V(\Delta)$. Это означает, что $L^{W'}_{\max}(\pi') \le 0$. Замечаем, что пример W' отличается от примера W (инверсного к V) тем, что все $\{r_j\}$ уменьшены на величину Δ. Если расписание π' "сдвинуть вправо" на Δ, то полученное расписание (π'') будет допустимым для примера W, причем $L^W_{\max}(\pi'') \le \Delta = L^V_{\max}(\pi_\tau)$. Так как требования в расписании π'' выполняются в порядке τ', то $L^W_{\max}(\pi_{\tau'}) \le L^W_{\max}(\pi'') \le L^V_{\max}(\pi_\tau)$.

[3]Если $\delta_B = 0$, то $\tilde{\pi}^B$ – оптимальное расписание для примера B.

Поменяв местами V с W, а τ с τ', получим обратное неравенство: $L^V_{\max}(\pi_\tau) \le L^W_{\max}(\pi_{\tau'})$, откуда вытекает искомое равенство $L^V_{\max}(\pi_\tau) = L^W_{\max}(\pi_{\tau'})$. □

Заметим, что если $L_{\max}(\pi_\tau)$ является минимальным значением для примера V, то $L_{\max}(\pi_{\tau'})$ будет минимальным для инверсного примера W. Отсюда вытекает следующее следствие леммы 1.3.

Следствие 1.2 *Если перестановка τ является оптимальной для примера V, то инверсная перестановка τ' будет оптимальной для примера, инверсного к V.*

Лемма 1.4 *Пусть пример C наследует у примера B продолжительности обслуживания и директивные сроки, а $\tilde{\pi}^C$ – приближённое решение примера C, удовлетворяющее условию*

$$L^C_{\max}(\tilde{\pi}^C) - L^C_{\max}(\pi^C) \le \delta_C. \tag{1.4}$$

Тогда

$$0 \le L^B_{\max}(\tilde{\pi}^C) - L^B_{\max}(\pi^B) \le \rho_r(B,C) + \delta_C.$$

Доказательство. Рассмотрим два инверсных примера E и F к примерам B и C, соответственно, с параметрами требований $\{r^E_j = -d^B_j, p^E_j = p^B_j, d^E_j = -r^B_j\}$ и $\{r^F_j = -d^C_j, p^F_j = p^C_j, d^F_j = -r^C_j\}$. Пусть τ^E и τ^F – инверсные перестановки для перестановок π^B и π^C, соответственно. Согласно Следствию 1.2, перестановки π^E и π^F – оптимальные для примеров E и F, соответственно. Тогда по лемме 1.2

$$\delta_C + \rho_d(E,F) \ge L^E_{\max}(\pi^E_{\tau^F}) - L^E_{\max}(\pi^E) \ge 0,$$

где

$$\rho_d(E,F) = \max_{j\in N}\left\{d^E_j - d^F_j\right\} + \max_{j\in N}\left\{d^F_j - d^E_j\right\}.$$

Согласно лемме 1.3, $L^B_{\max}(\pi^B) = L^E_{\max}(\pi^E)$ и $L^B_{\max}(\pi^B_{\tau^C}) = L^E_{\max}(\pi^E_{\tau^F})$. Следовательно,

$$\delta_C + \rho_d(E,F) \ge L^B_{\max}(\pi^B_{\tau^C}) - L^B_{\max}(\pi^B) \ge 0. \tag{1.5}$$

Имеем $\rho_d(E,F) = \max\limits_{j\in N}\{d^E_j - d^F_j\} + \max\limits_{j\in N}\{d^F_j - d^E_j\} = \max\limits_{j\in N}\{r^C_j - r^B_j\} + \max\limits_{j\in N}\{r^B_j - r^C_j\} = \rho_r(B,C)$. Отсюда и из (1.5) следует утверждение леммы.

□

Теорема 1.1 *Пусть пример C наследует у примера A длительности обслуживания требований. Тогда*

$$0 \le L^A_{\max}(\pi^A_{\tau^C}) - L^A_{\max}(\pi^A) \le \rho(A, C),$$

где $\rho(A, C) = \rho_d(A, C) + \rho_r(A, C)$.

Доказательство. Неравенство $0 \le L^A_{\max}(\pi^A_{\tau^C}) - L^A_{\max}(\pi^A)$ вытекает из оптимальности расписания π^A для примера A. Докажем неравенство $L^A_{\max}(\pi^A_{\tau^C}) - L^A_{\max}(\pi^A) \le \rho(A, C)$.

Рассмотрим пример $B = \{(r^B_j, p^B_j, d^B_j) \,|\, j \in N\}$ с параметрами требований $\{r^B_j = r^A_j, p^B_j = p^A_j = p^C_j, d^B_j = d^C_j\}$ и его оптимальное расписание π^B. По лемме 1.4 имеем

$$L^B_{\max}(\pi^B_{\tau^C}) - L^B_{\max}(\pi^B) \le \rho_r(B, C) = \rho_r(A, C). \tag{1.6}$$

Из оптимальности π^B для примера B

$$L^B_{\max}(\pi^B) \le L^B_{\max}(\pi^B_{\tau^A}). \tag{1.7}$$

Из леммы 1.1 для примеров A и B и расписаний π^C и π^A

$$L^A_{\max}(\pi^A_{\tau^C}) - L^B_{\max}(\pi^B_{\tau^C}) \le \max_{j \in N}\{d^B_j - d^A_j\}, \tag{1.8}$$

$$L^B_{\max}(\pi^B_{\tau^A}) - L^A_{\max}(\pi^A) \le \max_{j \in N}\{d^A_j - d^B_j\}. \tag{1.9}$$

Сложив неравенства (1.6)–(1.9), получим

$$L^A_{\max}(\pi^A_{\tau^C}) - L^A_{\max}(\pi^A) \le \rho_r(A, C) + \rho_d(A, B) = \rho_r(A, C) + \rho_d(A, C) = \rho(A, C).$$

Теорема доказана. □

Следствие 1.3 $0 \le L^C_{\max}(\pi^C_{\tau^A}) - L^C_{\max}(\pi^C) \le \rho(A, C)$. □

Таким образом, если после решения примера задачи $1 \mid r_j \mid L_{\max}$ изменились времена поступления и/или директивные сроки требований, то мы можем оценить абсолютную погрешность расписания, соответствующего оптимальной перестановке уже решенного примера, не решая сам измененный пример.

Рассмотрим в пространстве всех примеров задачи размерности n подпространство Φ^n_p, в котором примеры имеют одинаковые продолжительности обслуживания $p_1 = p_2 = \ldots = p_n = p$. Можно убедиться, что функция

$\rho(A,B)$ удовлетворяет свойствам *псевдометрики* для пространства Φ_p^n. Очевидно, что $\rho(A,B) = \rho(B,A)$ и $\rho(A,B) = 0$ если $A = B$. Докажем, что выполняется неравенство треугольника: $\rho(A,C) \le \rho(A,B) + \rho(B,C)$. Имеем

$$\begin{aligned}\max_{j\in N}\{d_j^A - d_j^C\} =& d_{j'}^A - d_{j'}^C = d_{j'}^A - d_{j'}^B + d_{j'}^B - d_{j'}^C \le \\ \le& \max_{j\in N}\{d_j^A - d_j^B\} + \max_{j\in N}\{d_j^B - d_j^C\}.\end{aligned} \tag{1.10}$$

Для остальных слагаемых $\max\limits_{j\in N}\{d_j^C - d_j^A\}$, $\max\limits_{j\in N}\{r_j^A - r_j^C\}$ и $\max\limits_{j\in N}\{r_j^C - r_j^A\}$ функции $\rho(A,C)$ неравенства вида (1.10) доказываются аналогично.

Функция $\rho(A,B)$ не является метрикой, так как для разных примеров $A, B \in \Phi_p^n$ может выполняться $\rho(A,B) = 0$, однако $\rho(A,B)$ можно рассматривать как расстояние между примерами. Согласно теореме 1.1, если $A \ne B$, $A, B \in \Phi_p^n$, и $\rho(A,B) = 0$, то примеры A и B имеют одинаковые множества оптимальных расписаний, то есть являются *эквивалентными*. То есть, расстояние между примерами из пространства Φ_p^n равняется нулю тогда и только тогда, когда примеры эквивалентны между собой.

Далее, в последующих разделах данной главы будет показано, что и продолжительности обслуживания требований тоже "можно менять" и получится "полноценная" метрика.

1.4 Схема нахождения приближённого решения

Результат, полученный в разделе 1.3, может быть использован для приближённого решения произвольного примера задачи $1 \mid r_j \mid L_{\max}$. В данном разделе рассматривается схема нахождения приближённого решения с использованием алгоритмов, разработанных для полиномиально разрешимых специальных случаев задачи. Далее будут представлены варианты применения предложенной схемы: в подразделе 1.4.1 – на основе случая $1 \mid d_i \le d_j$, $d_i - r_i - p_i \ge d_j - r_j - p_j \mid L_{\max}$ [24], в подразделе 1.4.2 – на основе случая Хогевена [130]. Для обоих вариантов будут найдены аналитические формулы для оценки сверху погрешности получаемого решения. Наконец, в подразделе 1.5.2 экспериментальным путём будут оценены соотношения среднего фактического значения абсолютной погрешности к её теоретической оценке сверху.

Заметим, что по нашим сведениям в литературе встречается только один пример полиномиального алгоритма – алгоритма Шраге (Schrage [192]), который гарантирует абсолютную погрешность полученного решения. Мак-

симальная погрешность полученного алгоритмом Шраге решения равняется максимальной продолжительности обслуживания $\max\limits_{j\in N} p_j$.

Рассмотрим произвольный пример A задачи $1 \mid r_j \mid L_{\max}$ с параметрами требований $\{r_j^A, p_j^A, d_j^A\}$, $j \in N$. Идея схемы приближённого решения примера A заключается в изменении времён поступления и/или директивных сроков требований примера A таким образом, чтобы получившийся в результате этого пример C с параметрами требований $\{r_j^C, p_j^C = p_j^A, d_j^C\}$, $j \in N$, принадлежал известному полиномиально разрешимому классу примеров задачи. В этом случае для решения примера C мы можем использовать полиномиальный алгоритм, а затем найденную им оптимальную перестановку требований применить к примеру A. Тогда, согласно теореме 1.1, абсолютная погрешность решения не будет превосходить расстояния $\rho(A, C)$ между примерами A и C.

Затем в задачу линейного программирования будут "добавлены"и продолжительности обслуживания требований.

Естественно, что при сведении примера A к примеру C значение $\rho(A, C)$ должно быть минимально. Таким образом, задача сводится к отысканию для заданного примера A наиболее близкого примера из заданного полиномиально разрешимого класса, т.е. необходимо спроецировать пример (точку) A на полиноминально разрешимую область.

Рассмотрим случай, когда некоторый полиномиально разрешимый класс примеров нашей задачи определяется системой линейных неравенств вида:

$$\mathbf{A}R^C + \mathbf{B}P^C + \mathbf{C}D^C \le H, \tag{1.11}$$

(при ограничениях $p_j^C \ge 0,\ j \in N$),
где $R^C = (r_1^C, \ldots, r_n^C)^T$, $P^C = (p_1^C, \ldots, p_n^C)^T$, $D^C = (d_1^C, \ldots, d_n^C)^T$, $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ – матрицы размерности $m \times n$, и $H = (h_1, \ldots, h_m)^T$ – m-мерный вектор, где верхний индекс ^{T} обозначает транспонирование, т.е. система неравенств (1.11) содержит m неравенств.

Необходимо заметить, что все полиноминально разрешимые случаи задачи $1 \mid r_j \mid L_{\max}$ описываются некоторой системой линейных неравенств вида (1.11).

Гипотеза: для любой NP—трудной задачи теории расписаний все полиноминально разрешимые случаи задачи можно задать некоторой системой линейных неравенств вида (1.11).

Тогда для отыскания в этом классе примера C, минимизирующего расстояние $\rho(A, C)$, достаточно решить следующую задачу линейного програм-

мирования:

$$\begin{cases} (x_d - y_d) + (x_r - y_r) \longrightarrow \min \\ y_d \le d_j^A - d_j^C \le x_d, \quad j \in N; \\ y_r \le r_j^A - r_j^C \le x_r, \quad j \in N; \\ p_j^C = p_j^A, \quad j \in N; \\ \mathbf{A}R^C + \mathbf{B}P^C + \mathbf{C}D^C \le H. \end{cases} \tag{1.12}$$

При решении задачи (1.12) необходимо учитывать специфику матриц $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ и H.

Например, для класса примеров, для которых $d_j = r_j + p_j$, $j \in N$, в системе линейных ограничений (1.11) матрицы $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ и H задаются следующим образом:

$$\mathbf{A} = \mathbf{B} = (\mathbf{I} \oplus (-\mathbf{I}))^T, \quad \mathbf{C} = ((-\mathbf{I}) \oplus \mathbf{I})^T, \quad H = (h)^T,$$

где $\mathbf{I}$ – единичная $n \times n$-матрица, $h = (0, \ldots, 0) \in \mathbb{R}^{2n}$, $\mathbf{F} \oplus \mathbf{G}$ обозначает конкатенацию матриц $\mathbf{F}$ (размером $l \times p$) и $\mathbf{G}$ (размером $l \times q$).

В качестве следующего примера возьмем класс примеров задачи $1 \mid r_j \mid L_{\max}$, когда $d_j = \delta$, $j \in N$, где δ – константа. Оптимальным расписанием для примеров данного класса является расписание, при котором требования упорядочены по неубыванию моментов поступления. Обозначим его как расписание π'. Такое расписание может быть найдено за $O(n \log n)$ операций, когда требования упорядочиваются в порядке неубывания моментов поступления на обслуживание r_j. Чтобы свести произвольный пример A задачи к примеру C из рассматриваемого класса, необходимо приравнять все директивные сроки к какой-либо константе δ. Сформулируем задачу линейного программирования, аналогичную (1.12), для нахождения примера C с минимальным расстоянием $\rho(A, C)$:

$$\begin{cases} x_d - y_d \longrightarrow \min \\ y_d \le d_j^A - \delta \le x_d, \quad j \in N. \end{cases} \tag{1.13}$$

Решением задачи (1.13) будет произвольное значение δ и x_d, y_d, такие что $x_d - y_d = \max\limits_{j \in N} d_j^A - \min\limits_{j \in N} d_j^A = \rho(A, C)$. То есть расстояние $\rho(A, C)$ в данном случае не зависит от того, к какой константе δ были сведены директивные сроки требований примера A. Таким образом, абсолютная погрешность расписания π' всегда будет не больше разницы между максимальным и минимальным директивными сроками требований примера A.

В случае, когда $r_j = \delta$, $j \in N$, где δ – константа, расписание π_{EDD} построенное по неубыванию директивных сроков (известный алгоритм Джек-

сона) за $O(n \log n)$ операций, будет оптимальным. Предлагаем читателю доказать, что

$$L^A_{\max}(\pi_{EDD}) - L^A_{\max}(\pi^A) \leq \max_{j \in N} r^A_j - \min_{j \in N} r^A_j$$

для любого примера A.

В следующих подразделах будут рассмотрены другие варианты применения предложенной схемы, которые не основаны на решении задачи (1.12).

1.4.1 Вариант схемы на основе случая $1 \mid d_i \leq d_j, d_i - r_i - p_i \geq d_j - r_j - p_j \mid L_{\max}$

Рассмотрим полиномиально разрешимый класс примеров исследуемой задачи $1 \mid r_j \mid L_{\max}$ [24]. Пример принадлежит к классу задач $1 \mid d_i \leq d_j, d_i - r_i - p_i \geq d_j - r_j - p_j \mid L_{\max}$, если он удовлетворяет свойству.

Свойство Ł. Будем говорить, что пример $C = \{(r^C_j, p^C_j, d^C_j) \mid j \in N\}$ удовлетворяет *свойству Ł*, если существует нумерация требований $\{1, 2, \ldots, n\}$, при которой выполняются соотношения

$$d^C_1 \leq \ldots \leq d^C_n; \quad \Delta^C_1 \geq \ldots \geq \Delta^C_n, \tag{1.14}$$

где $\Delta^C_j = d^C_j - r^C_j - p^C_j$ обозначает *временно́й запас* требования j.

Для случая (1.14) выполняется

$$\mathbf{A}R + \mathbf{A}P - \mathbf{A}D \leq 0,$$

$$\mathbf{A}D \leq 0,$$

где матрица $\mathbf{A}$, $(n-1) \times n$

$$\mathbf{A} = \begin{pmatrix} 1 & -1 & 0 & 0 \ldots 0 & 0 & 0 \\ 0 & 1 & -1 & 0 \ldots 0 & 0 & 0 \\ 0 & 0 & 1 & -1 \ldots 0 & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & 0 \ldots 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \ldots 0 & 1 & -1 \end{pmatrix}.$$

Далее, для заданного примера A определим функцию

$$\rho^L(A) = \max_{i,j \in N} \rho^L_{ij}(A), \tag{1.15}$$

где

$$\rho^L_{ij}(A) = \min\{d^A_j - d^A_i, \Delta^A_j - \Delta^A_i\}. \tag{1.16}$$

Так как $\rho_{ii}(A) = 0$ для любого $i \in N$, то нетрудно видеть, что $\rho^L(A) \geq 0$ для любого примера A. Причем, $\rho^L(A) = 0$ тогда и только тогда, когда A принадлежит классу задач $1 \mid d_i \leq d_j, d_i - r_i - p_i \geq d_j - r_j - p_j \mid L_{\max}$. Таким образом, функцию $\rho^L(A)$ можно считать *мерой невыполнения* свойства Ł.

Для точного решения задачи $1 \mid r_j \mid L_{\max}$ для примеров из класса задач $1 \mid d_i \leq d_j, d_i - r_i - p_i \geq d_j - r_j - p_j \mid L_{\max}$ построен алгоритм сложности $O(n^3 \log n)$ операций [28].

Пусть дан пример рассматриваемой задачи A с параметрами требований $\{r_j^A, p_j^A, d_j^A\}$, $j \in N$, которые не принадлежит классу задач $1 \mid d_i \leq d_j, d_i - r_i - p_i \geq d_j - r_j - p_j \mid L_{\max}$. Применим для данного примера предложенную раннее схему нахождения приближённого решения. Сведём пример A к примеру C, который наследует у примера A продолжительности обслуживания и принадлежит классу задач $1 \mid d_i \leq d_j, d_i - r_i - p_i \geq d_j - r_j - p_j \mid L_{\max}$.

Следующая теорема позволяет найти необходимый пример C, для которого расстояние $\rho(A, C)$ будет минимальным.

Теорема 1.2 *Для любого примера A задачи $1|r_j|L_{\max}$, не принадлежащего классу задач $1 \mid d_i \leq d_j, d_i - r_i - p_i \geq d_j - r_j - p_j \mid L_{\max}$, и для любого примера C, наследующего длительности обслуживания примера A и принадлежащего данному классу, справедливо:*

$$\rho(A, C) \geq \rho^L(A). \tag{1.17}$$

Оценка (1.17) достигается на некотором примере C, который может быть найден за время $O(n \log n)$ операций.

Доказательство. Для любого примера C, принадлежащего классу задач $1 \mid d_i \leq d_j, d_i - r_i - p_i \geq d_j - r_j - p_j \mid L_{\max}$, параметры любой пары требований $i, j \in N$ должны удовлетворять одному из следующих неравенств:

$$d_i^C - d_j^C \geq 0 \tag{1.18}$$

или

$$\Delta_i^C - \Delta_j^C \geq 0. \tag{1.19}$$

Если для пары (i, j) выполняется (1.18), то из определения расстояния $\rho(A, C)$ получим

$$\rho(A, C) \geq \rho_d(A, C) \geq (d_j^A - d_j^C) + (d_i^C - d_i^A) \geq d_j^A - d_i^A. \tag{1.20}$$

В противном случае из (1.19) и определения функции $\rho(A, C)$ следуют соотношения

$$\begin{aligned}\rho(A,C)\geq&(d_j^A-d_j^C)+(r_j^C-r_j^A)+(d_i^C-d_i^A)+(r_i^A-r_i^C)=\\ =&(d_j^A-d_j^C-p_j)+(r_j^C-r_j^A-p_j)+(d_i^C-d_i^A-p_i)+\\ &+(r_i^A-r_i^C-p_i)=\\ =&\Delta_j^A-\Delta_i^A+\Delta_i^C-\Delta_j^C\geq\Delta_j^A-\Delta_i^A.\end{aligned}\tag{1.21}$$

Используя определения (1.15) и (1.16), из (1.20) и (1.21) имеем

$$\rho^L(A)=\max_{(i,j)\in N\times N}\rho_{ij}^L(A)\leq\rho(A,C),$$

что доказывает (1.17).

Для доказательства достижимости оценки (1.17) построим пример C, наследующий у примера A длительности и моменты поступления требований. В силу того, что параметры p_j и r_j не будут различаться для примеров A и C, мы будем обозначать их без верхнего индекса "A" и "C" соответственно.

Пронумеруем требования примера A по неубыванию величин r_j+p_j:

$$r_1+p_1\leq\ldots\leq r_n+p_n.\tag{1.22}$$

Определим возрастающую последовательность индексов $0=j_0<j_1<\ldots<j_K=n$ с помощью следующего простого алгоритма:
$j_0:=0;\ k:=0;$
while $j_k<n$ **do** $\quad\{k:=k+1;\ j_k:=\arg\min\limits_{\{j\,|\,j_{k-1}<j\leq n\}}d_j^A\}.$

Нетрудно видеть, что $K\leq n$ и

$$d_{j_1}^A\leq d_{j_2}^A\leq\ldots\leq d_{j_K}^A.\tag{1.23}$$

Положим

$$d_j^C=\begin{cases}d_{j_1}^A, & \text{при } j\leq j_1,\\ \min\{d_{j_k}^A,d_j^A-\Delta_j^A+\Delta_{j_{k-1}}^C\}, & \text{при } j_{k-1}<j\leq j_k,\ k>1.\end{cases}\tag{1.24}$$

Из (1.22)–(1.24) вытекает упорядоченность величин $\{d_j^C\}$ по неубыванию:

$$d_1^C\leq\ldots\leq d_n^C.\tag{1.25}$$

Покажем, что так определенный пример C принадлежит нашему классу задач $1\mid d_i\leq d_j, d_i-r_i-p_i\geq d_j-r_j-p_j\mid L_{\max}$. Ввиду (1.25), для этого достаточно доказать, что при любом $k=1,\ldots,K$ выполнены соотношения:

$$\Delta_i^C\geq\Delta_j^C,\ \forall\ i,j\in N\ i<j\leq j_k.\tag{1.26}$$

Неравенство (1.26) будем доказывать индукцией по k.

При $k = 1$ оно следует из (1.22) и (1.24):

$$\Delta_i^C = d_i^C - r_i - p_i = d_{j_1}^A - r_i - p_i \geq d_{j_1}^A - r_j - p_j = d_j^C - r_j - p_j = \Delta_j^C.$$

Пусть (1.26) верно при $k = k' - 1 \geq 1$. Докажем его для $k = k'$. Для этого достаточно рассмотреть лишь два случая:

а) $i \leq j_{k'-1} < j \leq j_{k'}$ и

б) $j_{k'-1} < i < j \leq j_{k'}$.

В случае а) по индукционному предположению имеем:

$$\begin{aligned}\Delta_i^C \geq \Delta_{j_{k'-1}}^C &\geq \min\{\Delta_{j_{k'-1}}^C, d_{j_{k'}}^A - d_j^A + \Delta_j^A\} \\ &= \min\{\Delta_{j_{k'-1}}^C + d_j^A - \Delta_j^A, d_{j_{k'}}^A\} - d_j^A + \Delta_j^A \\ &= d_j^C - r_j - p_j = \Delta_j^C.\end{aligned}$$

В случае б), применяя поочередно (1.24),(1.22),(1.24), получим:

$$\begin{aligned}\Delta_i^C &= d_i^C - r_i - p_i = \min\{d_{j_{k'}}^A - r_i - p_i, \Delta_{j_{k'-1}}^C\} \\ &\geq \min\{d_{j_{k'}}^A - r_j - p_j, \Delta_{j_{k'-1}}^C\} = d_j^C - r_j - p_j = \Delta_j^C.\end{aligned}$$

Таким образом, C принадлежит классу задач $1 \mid d_i \leq d_j, d_i - r_i - p_i \geq d_j - r_j - p_j \mid L_{\max}$.

Для доказательства достижимости (1.17) вначале покажем, что

$$d_j^C \leq d_j^A, \ \forall\, j. \tag{1.27}$$

Действительно, при $j \leq j_1$ по выбору j_1 имеем $d_j^C = d_{j_1}^A \leq d_j^A$.

При $j_{k-1} < j \leq j_k, \ k \geq 2$, из (1.24) имеем $d_j^C \leq d_{j_k}^A \leq d_j^A$ – по выбору j_k. Таким образом, (1.27) доказано.

Далее заметим, что если $\rho_{ij}^L(A)$, определяемое согласно (1.16), записать в виде $\rho_{ij}^L(A) = \min\{d_j^A - d_i^A, d_j^A - d_i^A + (r_i + p_i) - (r_j + p_j)\}$, то ввиду (1.22) будем иметь:

$$\rho_{ij}^L(A) = \begin{cases} d_j^A - d_i^A, & \text{при } i \geq j, \\ \Delta_j^A - \Delta_i^A, & \text{при } i < j. \end{cases} \tag{1.28}$$

Из (1.24) при $k \geq 2$ и $j = j_k$ имеем $d_{j_k}^C = \min\{d_{j_k}^A, d_{j_k}^A - \Delta_j^A + \Delta_{j_{k-1}}^C\}$. Вычтя из обеих частей этого равенства по $(r_{j_k} + p_{j_k})$, получим

$$\Delta_{j_k}^C = \min\{\Delta_{j_k}^A, \Delta_{j_{k-1}}^C\}. \tag{1.29}$$

Из (1.29) и равенства $\Delta^C_{j_1} = \Delta^A_{j_1}$ получаем свойство

$$\forall k \geq 1, \exists \nu \leq k : \Delta^C_{j_k} = \Delta^A_{j_\nu}. \tag{1.30}$$

Далее для любого j докажем неравенство

$$d^A_j - d^C_j \leq \rho^L(A). \tag{1.31}$$

При $j \leq j_1$ имеем $d^A_j - d^C_j = d^A_j - d^A_{j_1} \overset{\text{из (1.28)}}{=} \rho^L_{j_1 j}(A) \leq \rho^L(A)$.

При $j_{k-1} < j \leq j_k,\ k \geq 2$, имеем

$$\begin{aligned} d^A_j - d^C_j = & d^A_j - \min\{d^A_{j_k}, d^A_j - \Delta^A_j + \Delta^C_{j_{k-1}}\} = \max\{d^A_j - d^A_{j_k}, \Delta^A_j - \Delta^C_{j_{k-1}}\} = \\ & \text{(с учётом (1.30) для некоторого } j_\nu \leq j_{k-1}) \\ = & \max\{d^A_j - d^A_{j_k}, \Delta^A_j - \Delta^A_{j_\nu}\} \overset{\text{из (1.28)}}{=} \max\{\rho^L_{j_k j}(A), \rho^L_{j_\nu j}(A)\} \leq \rho^L(A), \end{aligned}$$

что доказывает (1.31).

Наконец, поскольку $\rho_r(A, C) = 0$, имеем

$$\rho(A, C) = \rho_d(A, C) = \max_j\{d^A_j - d^C_j\} + \max_j\{d^C_j - d^A_j\} \overset{\text{из (1.31) и (1.27)}}{\leq} \rho^L(A),$$

откуда, ввиду (1.17), получаем равенство $\rho(A, C) = \rho^L(A)$. Для доказательства оценки трудоёмкости алгоритма нахождения искомого примера C целесообразно пронумеровать требования примера A в обратном порядке – по невозрастанию величин $r_j + p_j$, после чего применить следующий алгоритм линейной трудоёмкости для нахождения *семейства разделяющих индексов* $\{j_1, \ldots, j_K\}$:

Теорема 1.2 доказана. □

Алгоритм 1.1 Алгоритм нахождения *семейства разделяющих индексов*

```
1: k := 1; j_1 := 1; i := 1;
2: for i := 2 to n do
3:    if d_i^A < d_{j_k}^A then
4:       k := k + 1; j_k := i
5:    end if
6: end for
```

На основе теоремы 1.2 представим алгоритм 1.2, который сводит произвольный пример A к примеру C, для которого (1.17) выполняется как равенство. В алгоритме предполагается, что требования в примере A отсортированы по невозрастанию величин $r^A_j + p^A_j$. Заметим, что нумерация требований,

используемая в алгоритме, обратна нумерации, используемой в доказательстве теоремы 1.2.

Фактически пример C (точка в $3n$—мерном пространстве) является проекцией исходного примера A (также точка в $3n$—мерном пространстве) на полиноминально разрешимую область в некоторой метрике.

Пример. Рассмотрим работу алгоритма 1.2 на примере. Пусть нам дан пример A задачи $1 \mid r_j \mid L_{\max}$ с параметрами требований, указанными в таблице 1.1. Как и необходимо, требования отсортированы по невозрастанию $r_j^A + p_j^A$.

Алгоритм 1.2 Алгоритм построения примера, удовлетворяющего разрешимому случаю $1 \mid d_i \le d_j, d_i - r_i - p_i \ge d_j - r_j - p_j \mid L_{\max}$, минимизирующий $\rho(A, C)$

for $j := 1$ **to** n **do**
  $r_j^C := r_j^A$; $p_j^C := p_j^A$
end for
$k := 1$; $j_1 = 1$
for $i := 2$ **to** n **do**
  if $d_i^A < d_{j_k}^A$ **then**
    $k := k + 1$; $j_k := i$
  end if
end for
for $i := n$ **downto** j_k **do**
  $d_i^C := d_{j_k}^A$
end for
while $k > 1$ **do**
  for $i := j_k - 1$ **downto** j_{k-1} **do**
    $d_i^C := \min\{d_{j_{k-1}}^A, d_j^A - \Delta_j^A + \Delta_{j_k}^C\}$
  end for
  $k := k - 1$
end while

Таблица 1.1: Параметры требований для примера A

$j =$	1	2	3	4	5	6	7	8
r_j^A	7	5	3	5	1	2	3	0
p_j^A	2	4	5	3	5	3	1	4
d_j^A	16	18	13	14	15	11	12	14

Начинается исполнение алгоритма. $N = \{1, \ldots, 8\}$. После первого цикла получаем $r_j^C := r_j^A$ и $p_j^C := p_j^A$, $\forall j \in N$. После второго цикла имеем: $j_1 = 1$, $j_2 = 3$, $j_3 = 6$.

После третьего цикла получаем $d_6^C = d_7^C = d_8^C = d_6^A = 11$.

Начнем выполнение цикла WHILE.

$k = 3$.

$$d_5^C = \min\{d_3^A, d_6^C - r_6^C - p_6^C + r_5^A + p_5^A\} = 12.$$
$$d_4^C = \min\{d_3^A, d_6^C - r_6^C - p_6^C + r_4^A + p_4^A\} = 13.$$
$$d_3^C = \min\{d_3^A, d_6^C - r_6^C - p_6^C + r_3^A + p_3^A\} = 13.$$

$k = 2$.

$$d_2^C = \min\{d_1^A, d_3^C - r_3^C - p_3^C + r_2^A + p_2^A\} = 14.$$
$$d_1^C = \min\{d_1^A, d_3^C - r_3^C - p_3^C + r_1^A + p_1^A\} = 14.$$

Алгоритм заканчивает работу. Итак, мы получили пример C, принадлежащий классу $1 \mid d_i \leq d_j, d_i - r_i - p_i \geq d_j - r_j - p_j \mid L_{\max}$. Параметры требований примера C указаны в таблице 1.2. Найдём значение абсолютной погрешности в данном случае: $\rho(A, C) = \max\limits_{j \in N}\{d_j^A - d_j^C\} + \max\limits_{j \in N}\{d_j^C - d_j^A\} = d_2^A - d_2^C + 0 = 4$. Процесс работы алгоритма 1.2 также проиллюстрирован на рисунке 1.1.

Таблица 1.2: Параметры требований для найденного примера C

$j =$	1	2	3	4	5	6	7	8
r_j^C	7	5	3	5	1	2	3	0
p_j^C	2	4	5	3	5	3	1	4
d_j^C	14	14	13	13	12	11	11	11

1.4.2 Вариант схемы на основе случая Хогевена

Следующий вариант схемы приближённого решения основан на сведении заданного примера к примеру из полиномиально разрешимого класса Хогевена [130].

Будем говорить, что пример $C = \{(r_j^C, p_j^C, d_j^C) \mid j \in N\}$ принадлежит классу Хогевена, если существует такая константа β, что:

$$r_j^C \in [d_j^C - p_j^C - \beta, d_j^C - \beta], \qquad \forall j \in N. \tag{1.32}$$

Для точного решения примеров из класса Хогевена существует алгоритм сложности $O(n^2 \log n)$ операций [130]. Ограничения (1.32) можно также переписать следующим образом. Существует такая константа β, что:

$$\beta \in [d_j^C - p_j^C - r_j^C, d_j^C - r_j^C] \qquad \forall j \in N. \tag{1.33}$$

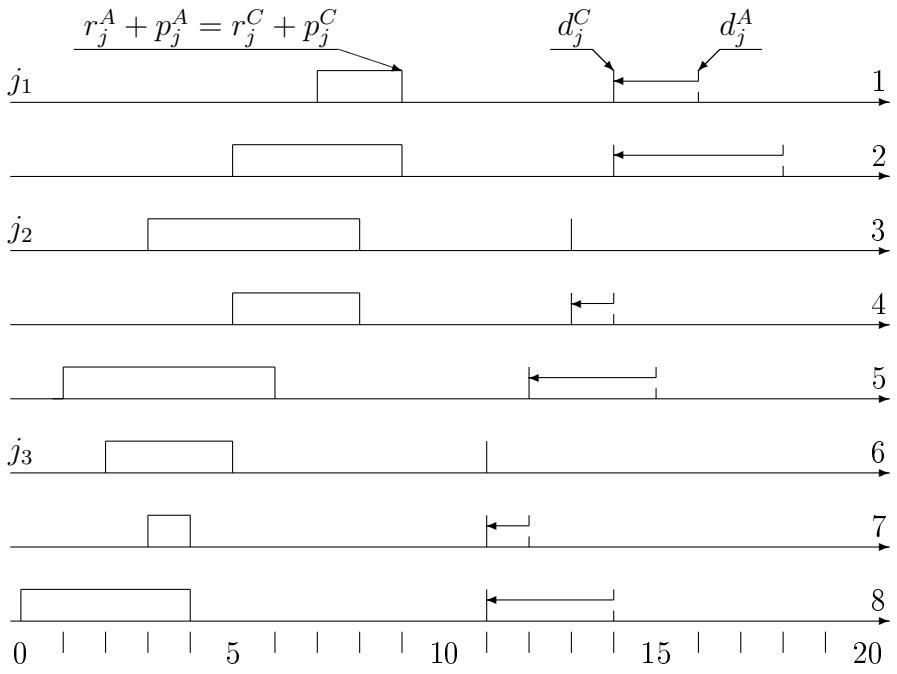

Рис. 1.1: Работа алгоритма 1.2 на примере

Условия (1.33) можно записать в виде неравенств (1.11) следующим образом:

$$\begin{aligned} -r_j - p_j + d_j &\leq \beta, \\ r_j - d_j &\leq -\beta, \end{aligned}$$

где β – константа, определяющая класс Хогевена.

Для заданного примера A определим функцию

$$\rho^H(A) = \max_{i,j \in N} \rho_{ij}^H(A), \tag{1.34}$$

где

$$\rho_{ij}^H(A) = d_j^A - r_j^A - p_j^A - d_i^A + r_i^A. \tag{1.35}$$

Заметим, что $\rho^H(A) \leq 0$, тогда и только тогда, когда A принадлежит классу Хогевена. Если $\rho^H(A) > 0$, то существует такая пара требований $i', j' \in N$, что $d_{j'}^A - p_{j'}^A - r_{j'}^A > d_{i'}^A - r_{i'}^A$ и отрезки $[d_{i'}^A - p_{i'}^A - r_{i'}^A, d_{i'}^A - r_{i'}^A]$ и $[d_{j'}^A - p_{j'}^A - r_{j'}^A, d_{j'}^A - r_{j'}^A]$ не пересекаются. Следовательно, не найдется такой константы β, что выполняется (1.33).

Пусть дан пример A с параметрами требований $\{r_j^A, p_j^A, d_j^A\}$, $j \in N$, который не принадлежит классу Хогевена. Применим для данного примера предложенную ранее схему нахождения приближённого решения. Сведём пример A к примеру C, который наследует у примера A продолжительности обслуживания и принадлежит классу Хогевена.

Следующая теорема позволяет найти пример C, для которого расстояние $\rho(A, C)$ будет минимальным.

Теорема 1.3 *Для любого примера A задачи $1|r_j|L_{\max}$, не принадлежащего классу Хогевена, и для любого примера C, наследующего длительности обслуживания примера A и принадлежащего классу Хогевена, справедлива оценка расстояния между A и C:*

$$\rho(A,C) \geq \rho^H(A). \tag{1.36}$$

Оценка (1.36) достигается на некотором примере C, который может быть найден за время $O(n)$ операций.

Доказательство. Как это было отмечено выше, для любого примера C, принадлежащего классу Хогевена, параметры любой пары требований $i,j \in N$ должны удовлетворять следующему неравенству:

$$0 \leq -\rho^H_{ij}(C) = d^C_i - r^C_i - d^C_j + p^C_j + r^C_j. \tag{1.37}$$

Добавим в обе части неравенства (1.37) $\rho^H_{ij}(A)$, учитывая, что $p^C_j = p^A_j$. Используя (1.35) и определение расстояния $\rho(A,C)$, получим

$$\rho^H_{ij}(A) \leq (d^A_j - d^C_j) + (r^C_j - r^A_j) + (d^C_i - d^A_i) + (r^A_i - r^C_i) \leq \rho(A,C). \tag{1.38}$$

Так как (1.38) выполняется для любой пары требований $i,j \in N$, из (1.34) имеем

$$\rho^H(A) = \max_{i,j\in N} \rho^H_{ij}(A) \leq \rho(A,C).$$

Неравенство (1.36) доказано.

Для доказательства достижимости оценки (1.36) на некотором примере C построим пример C, наследующий у примера A длительности обслуживания и моменты поступления требований. В силу того, что параметры p_j и r_j не будут различаться для примеров A и C, мы будем обозначать их без верхнего индекса "A" и "C" соответственно.

Обозначим $j^* = \arg\max_{j\in N}\{d^A_j - r^A_j - p^A_j\}$. Положим

$$d^C_j = \max\{d^A_{j^*} - r_{j^*} - p_{j^*} + r_j, d^A_j\} \quad \forall j \in N. \tag{1.39}$$

Рассмотрим произвольное требование $j \in N$. Если $d^A_j > d^A_{j^*} - r_{j^*} - p_{j^*} + r_j$, то $d^C_j = d^A_j$ и $d^C_j - d^A_j = 0 \leq \rho^H(A)$. Если $d^A_j \leq d^A_{j^*} - r_{j^*} - p_{j^*} + r_j$, то $d^C_j - d^A_j = d^A_{j^*} - r_{j^*} - p_{j^*} - d^A_j + r_j = \rho^H_{jj^*}(A) \leq \rho^H(A)$.

Отметим, что $d^C_{j^*} = d^A_{j^*}$, так как $d^A_{j^*} - r_{j^*} > d^A_{j^*} - r_{j^*} - p_{j^*}$. В итоге имеем $\max_{j\in N}\{d^A_j - d^C_j\} = d^A_{j^*} - d^C_{j^*} = 0$, $\max_{j\in N}\{d^C_j - d^A_j\} \leq \rho^H(A)$, $r_A - r_C = 0$,

$\forall j \in N$. Складывая $\max\limits_{j\in N}\{d_j^A - d_j^C\}$, $\max\limits_{j\in N}\{d_j^C - d_j^A\}$, $\max\limits_{j\in N}\{r_j^A - r_j^C\}$ и $\max\limits_{j\in N}\{r_j^C - r_j^A\}$ получаем, что $\rho(A,C) \le \rho^H(A)$. Отсюда и из (1.36) получаем $\rho(A,C) = \rho^H(A)$.

Теперь покажем, что построенный пример C удовлетворяет свойству Хогевена. Возьмём константу $\beta = d_{j^*}^C - r_{j^*} - p_{j^*} = d_{j^*}^A - r_{j^*} - p_{j^*}$. Рассмотрим произвольное требование $j \in N$. Если $d_j^A - r_j > d_{j^*}^A - r_{j^*} - p_{j^*}$, то из (1.39), $d_j^C = d_j^A$ и $d_j^C - r_j > \beta$. Имеем также $d_j^C - r_j - p_j \le \beta$ по определению j^* и β. Если же $d_j^A - r_j \le d_{j^*}^A - r_{j^*} - p_{j^*}$, то из (1.39), $d_j^C - r_j = \beta$. Очевидно, что тогда $d_j^C - r_j^C - p_j^C < \beta$. Следовательно, пример C удовлетворяет свойству Хогевена.

Доказательство трудоёмкости алгоритма сведения примера A к примеру C следует из того, что для данного преобразования мы должны применить формулу (1.39) n раз. □

На основе теоремы 1.3 получаем очевидный алгоритм 1.3, который сводит пример A к примеру C, для которого (1.36) выполняется как равенство.

Алгоритм 1.3 Алгоритм построения примера, удовлетворяющего случаю Хогевена, минимизирующий $\rho(A,C)$

1: $j^* = \arg\max\limits_{j\in N}\{d_j^A - r_j^A - p_j^A\}$
2: **for** $j := 1$; **to** n **do**
3: $\quad r_j^C := r_j^A$;
4: $\quad p_j^C := p_j^A$;
5: $\quad d_j^C = \max\{d_{j^*}^A - r_{j^*}^A - p_{j^*}^A + r_j^A, d_j^A\}$
6: **end for**

Для примера A из таблицы 1.1 оценка абсолютной погрешности решения, полученного с помощью варианта схемы на основе случая Хогевена, составляет $\rho^H(A) = 1$.

Пример, удовлетворяющий условиям Хогевена.

$j = 1$	1	2	3	4	5	6	7	8
r_j^C	7	5	3	5	1	2	3	0
P_j^C	2	4	5	3	5	3	1	4
d_j^C	17	18	13	15	15	12	13	14

Как и в разделе 1.4.1, построенный пример C (точка) является проекцией исходного примера A (точки) на класс примеров Хогевена.

1.5 Экспериментальное исследование полиномиальных алгоритмов решения задачи $1 \mid r_j \mid L_{\max}$

В данном разделе мы экспериментально оценим эффективность использования полиномиальных алгоритмов автора [24, 28, 29] и Хогевена [130] для решения общего случая задачи. Также будет исследована фактическая абсолютная погрешность расписаний, полученных с помощью вариантов схемы приближённого решения задачи, представленных в разделе 1.4.

При проведении экспериментальных исследований одним из первостепенных вопросов является способ генерации тестовых примеров. Мы покажем, что множество Φ^n всех примеров задачи размерности n может быть преобразовано в ограниченное множество Ψ^n, где каждый элемент представляет собой бесконечное число эквивалентных примеров задачи. В подразделе 1.5.1 мы представим свою процедуру генерации и покажем, что она генерирует примеры по равномерному распределению на множестве Ψ^n.

Используя данную процедуру, в подразделе 1.5.2 мы оценим насколько теоретическая оценка абсолютной погрешности $\rho(A, C)$ превышает среднее практическое значение абсолютной погрешности приближённого решения, полученного по схеме, представленной в разделе 1.4. Сравнение теоретической и практической абсолютной погрешности будет произведено для вариантов схемы на основе случаев автора и Хогевена.

Далее в подразделе 1.5.3 экспериментальным путём определяется эффективность алгоритмов автора и Хогевена, а также алгоритма Шраге [192] при их использовании для решения общего случая задачи $1 \mid r_j \mid L_{\max}$.

1.5.1 Способ генерации примеров

Множество Φ^n всех примеров задачи $1 \mid r_j \mid L_{\max}$ с числом требований n может быть представлено частью $3n$-мерного пространства, где координаты каждой точки представляют собой параметры требований $\{r_j, p_j, d_j\}$, $p_j > 0$, $j \in N$. Множество Φ^n можно преобразовать в некоторое ограниченное множество Ψ^n.

Докажем сначала, что примеры с параметрами требований $\{r_j, p_j, d_j\}$ и $\{\alpha r_j, \alpha p_j, \alpha d_j\}$, где $\alpha > 0$, имеют одно и тоже множество оптимальных расписаний.

Теорема 1.4 *Пусть π^*, $\pi^* \in \Pi(N)$, – оптимальное расписание для примера P с параметрами требований $\{r_j, p_j, d_j\}$, $j \in N$, тогда расписание π^* будет*

также оптимальным для примера Q с параметрами требований $\{r_j^Q = \alpha r_j, p_j^Q = \alpha p_j, d_j^Q = \alpha d_j\}$, $j \in N$, *где* $\alpha \in \mathbb{R}$, $\alpha > 0$.

Доказательство. Пусть имеется произвольное раннее расписание π, $\pi \in \Pi(N)$. Перенумеруем требования в том порядке, в котором они обслуживаются в расписании π. Пусть $f_j^P(\pi) = \max\limits_{i \in N}\{i \mid i \leq j, s_i^P(\pi) = r_i^P\}$. Очевидно, что, если $f_j^P(\pi) \neq j$, то $s_j^P(\pi) = s_{j-1}^P + p_{j-1}$.

Докажем по индукции, что $s_j^Q(\pi) = \alpha s_j^P(\pi)$, $j \in N$.
Для требования 1 имеем: $s_1^Q(\pi) = r_1^Q = \alpha r_1^P = \alpha s_1^P(\pi)$. Пусть $s_i^Q(\pi) = \alpha s_i^P(\pi)$.

Если $f_{i+1}^P(\pi) = i+1$, то $s_{i+1}^Q(\pi) = r_{i+1}^Q = \alpha r_{i+1}^P = \alpha s_{i+1}^P(\pi)$, так как $r_{i+1}^Q = \alpha r_{i+1}^P \geq \alpha(s_i^P(\pi) + p_i^P) = s_i^Q(\pi) + p_i^Q$.

Если $f_{i+1}^P(\pi) \neq i+1$, то $s_{i+1}^Q(\pi) = s_i^Q(\pi) + p_i^Q = \alpha s_i^P(\pi) + \alpha p_i^P = \alpha s_{i+1}^P(\pi)$, так как $s_i^Q(\pi) + p_i = \alpha s_i^P(\pi) + \alpha p_i^P = \alpha s_{i+1}^P(\pi) \geq \alpha r_{i+1}^P = r_{i+1}^Q$.

Так как $s_j^Q(\pi) = \alpha s_j^P(\pi)$, $j \in N$, то $L_j^Q(\pi) = s_j^Q(\pi) + p_j^Q - d_j^Q = \alpha s_j^P(\pi) + \alpha p_j^P - \alpha d_j^P = \alpha L_j^P(\pi)$, $\forall j \in N, \forall \pi \in \Pi(N)$. Следовательно, если π^* – оптимальное расписание для примера P, то $\min\limits_{\pi \in \Pi(N)} \max\limits_{j \in N} L_j^Q(\pi) = \min\limits_{\pi \in \Pi(N)} \max\limits_{j \in N} \alpha L_j^P(\pi) = \max\limits_{j \in N} \alpha L_j^P(\pi^*) = \max\limits_{j \in N} L_j^Q(\pi^*)$ и π^* – оптимальное расписание для примера Q.

□

Вероятнее всего, данная теорема доказывалась ранее, но автор не нашёл ссылки на доказательство... Согласно теореме 1.4 каждая точка, соответствующая примеру с параметрами требований $\{r_1, \ldots, r_n, p_1, \ldots, p_n, d_1, \ldots, d_n\}$, может быть спроецирована на единичную сферу в $3n$-мерном пространстве без изменения множества оптимальных расписаний. Следовательно в качестве множества всех примеров задачи размерности n можно взять множество $\bar{\Psi}^n$, которое является частью единичной сферы в $3n$-мерном пространстве, где $p_j > 0$, $j \in N$. Таким образом, для каждого примера из множества Φ^n существует эквивалентный пример из множества $\bar{\Psi}^n$, для которого $\sqrt{r_j^2 + p_j^2 + d_j^2} = 1$.

В экспериментальном исследовании одним из оцениваемых параметров будет относительная погрешность найденного расписания. Поэтому, оптимальное значение целевой функции не должно равняться нулю. В противном случае значение относительной погрешности может равняться бесконечности. Во избежание подобного результата параметры требований тестовых примеров мы будем модифицировать таким образом, чтобы выполнялось $r_j \geq 0$ и $d_j \leq 0$, $j \in N$. Тогда временно́е смещение каждого требования, и, соот-

ветственно, максимального, будет больше нуля, а множество оптимальных расписаний останется неизменным.

Рассмотрим пример A с параметрами требований $\{r_j^A, p_j^A, d_j^A\}$, $j \in N$, и пример B с параметрами требований $\{r_j^B = r_j^A + \alpha, p_j^B = p_j^A, d_j^B = d_j^A + \beta\}$, $j \in N$, где α и β – некоторые константы $\alpha,\ \beta \in R$. Согласно теореме 1.1, $\rho(A, B) = 0$, из чего следует, что примеры A и B эквивалентны.

Преобразуем множество $\bar{\Psi}^n$ примеров задачи во множество Ψ^n. Для этого каждый пример A из $\bar{\Psi}^n$ преобразуем в эквивалентный ему пример B из Ψ^n следующим образом:

$$r_j^B = r_j^A - \min_{i \in N} r_i^A, \quad p_j^B = p_j^A, \quad d_j^B = d_j^A - \max_{i \in N} d_i^A, \quad \forall j \in N.$$

Таким образом, множества примеров $\bar{\Psi}^n$ и Ψ^n эквивалентны (множества оптимальных расписаний соответствующих примеров совпадают) и для рассмотрения множества всех примеров размерности n можно ограничиться примерами из множества Ψ^n, оптимальное значение целевой функции для которых строго положительно.

Следующий алгоритм 1.4 равномерно генерирует примеры из множества $\bar{\Psi}^n$ для заданного n, а затем преобразует их в примеры из множества Ψ^n. Так как $\bar{\Psi}^n$ представляет собой часть единичной $3n$-мерной сферы, для получения равномерной генерации примеров из $\bar{\Psi}^n$, параметры требований (координаты) генерируются по нормальному закону распределения.

1.5.2 Оценка экспериментального значения абсолютной погрешности

В данном исследовании будет найдено практическое значение параметра α. Данный параметр определяет отношение практического значения абсолютной погрешности к теоретической оценке в процентах. Параметр α вычисляется следующим образом:

$$\alpha = \sum_{i=1}^{K} \frac{(L_{\max}(\pi_i^C) - L_{\max}(\pi_i^A))}{\rho_i(A, C)} \cdot 100\%$$

Алгоритм 1.4 Алгоритм равномерной генерации примеров на множестве Ψ^n

1: генерируем параметры требований $\{\hat{r}_j, \hat{p}_j, \hat{d}_j\}$ согласно нормальному распределению с параметрами $m = 0$ и $\delta = 1$
2: **for all** $j \in N$ **do**
3: **if** $\hat{p}_j < 0$ **then**
4: $p_j := -\hat{p}_j$
5: **else**
6: $p_j := \hat{p}_j$
7: **end if**
8: $r_j := \hat{r}_j - \min\limits_{j \in N}\{\hat{r}_j\}$
9: $d_j := \hat{d}_j - \max\limits_{j \in N}\{\hat{d}_j\}$
10: **end for**
11: $\Delta := \sqrt{\sum\limits_{j \in N} r_j^2 + \sum\limits_{j \in N} p_j^2 + \sum\limits_{j \in N} d_j^2}$.
12: **for all** $j \in N$ **do**
13: $r_j := \frac{r_j}{\Delta}$; $p_j := \frac{p_j}{\Delta}$; $d_j := \frac{d_j}{\Delta}$
14: **end for**

Здесь K – общее число сгенерированных тестовых примеров; $L_{\max}(\pi_i^A)$ – оптимальное значение максимального временно́го смещения для примера A в эксперименте i; $L_{\max}(\pi_i^C)$ – значение максимального временно́го смещения расписания π^C в эксперименте i, полученного по схеме нахождения приближённого решения; $\rho_i(A, C)$ – теоретическая оценка абсолютной погрешности в эксперименте i. Обозначим α_L и α_H, как значение параметра α для случая автора и случая Хогевена, соответственно. Напомним, что для первого случая $\rho(A, C) = \rho^L(A)$, а для второго $\rho(A, C) = \rho^H(A)$.

Для проведения эксперимента примеры генерировались согласно способу, приведенному в подразделе 1.5.1. Для каждой размерности пространства $n \in \{2, \ldots, 100\}$ было сгенерировано 10000 примеров.

На рисунке 1.2 показаны графики зависимости параметров α_L и α_H от размерности примеров n. Можно заметить, что значения двух исследуемых параметров отличаются друг от друга незначительным образом. С увеличением размерности и α_L, и α_H растут и стабилизируются в районе 45%. Можно сделать вывод, то среднее практическое значение абсолютной погрешности не превышает половины теоретической оценки абсолютной погрешности ρ. Нужно также заметить, что значение параметра α_H заметно ниже α_L для больших размерностей исследуемых примеров.

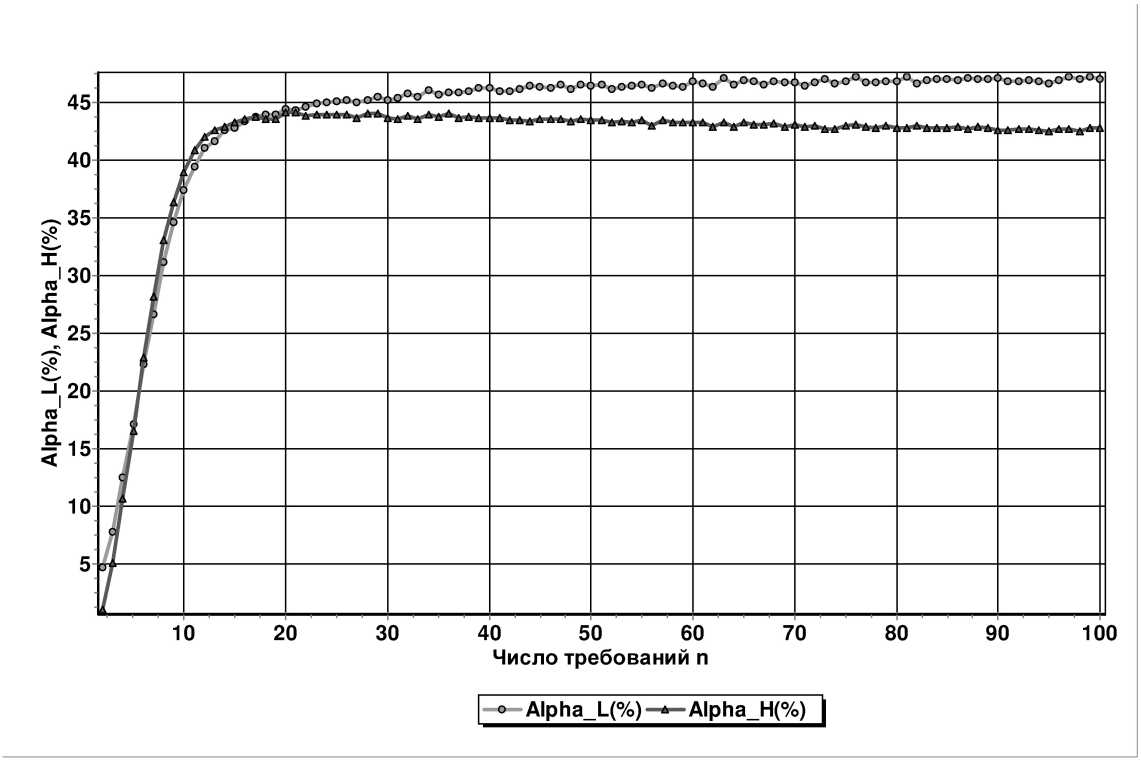

Рис. 1.2: Среднее отношение практической абсолютной погрешности к теоретической оценке

1.5.3 Эффективность применения полиномиальных алгоритмов для общего случая задачи

Эффективность алгоритмов Шраге, автора и Хогевена при решении произвольных примеров задачи $1 \mid r_j \mid L_{\max}$ мы будем оценивать с помощью следующих трех показателей.

- Параметр m определяет процент оптимально решенных алгоритмом примеров из числа сгенерированных и вычисляется по формуле:

$$m = \frac{K^*}{K} \cdot 100\%.$$

  Здесь K – общее число сгенерированных тестовых примеров, K^* – число оптимально решенных алгоритмом тестовых примеров.

- Параметры β_{av} и $\beta_{\max}$ определяют, соответственно, среднюю и максимальную относительную погрешность значения целевой функции найденного алгоритмом расписания для тестового примера. Погрешность определяется относительно оптимального значения целевой функции тестового примера. Параметры β_{av} и $\beta_{\max}$ вычисляются по следующим

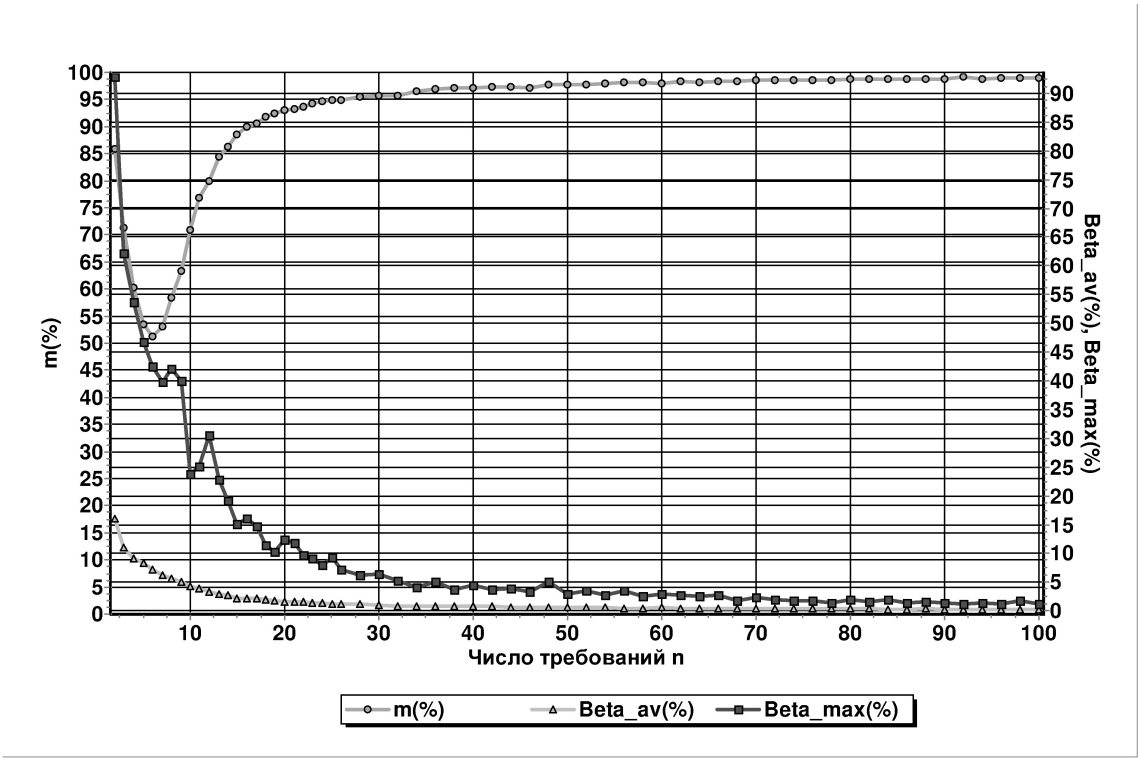

Рис. 1.3: Эффективность алгоритма Шраге

формулам:

$$\beta_{\text{av}} = \sum_{i=1}^{\overline{K}} \frac{(L_{\max}(\pi_i) - L_{\max}(\pi_i^*))}{L_{\max}(\pi_i^*)} \cdot 100\%;$$

$$\beta_{\max} = \max_{i=1,\ldots,\overline{K}} \left\{ \frac{L_{\max}(\pi_i) - L_{\max}(\pi_i^*)}{L_{\max}(\pi_i^*)} \cdot 100\% \right\}.$$

Здесь $\overline{K}$ – число сгенерированных примеров, для которых полученное алгоритмом решение не было оптимальным ($\overline{K} = K - K^*$), π_i и π_i^* – найденное алгоритмом расписание и оптимальное расписание для i-го сгенерированного примера, решённого неоптимально, соответственно.

В проведённом исследовании мы экспериментально определяли параметры m, β_{av} и $\beta_{\max}$ для трёх полиномиальных алгоритмов. Нами исследовались примеры с числом требований n от 2 до 100. Тестовые примеры генерировались по алгоритму 1.4 на $3n$-мерной единичной сфере. Для каждого исследуемого значения n было построено 10000 примеров.

На рисунке 1.3 представлены графики зависимости исследуемых параметров от числа требований n для алгоритма Шраге, на рисунке 1.4 – для алгоритма автора, и на рисунке 1.5 – для алгоритма Хогевена.

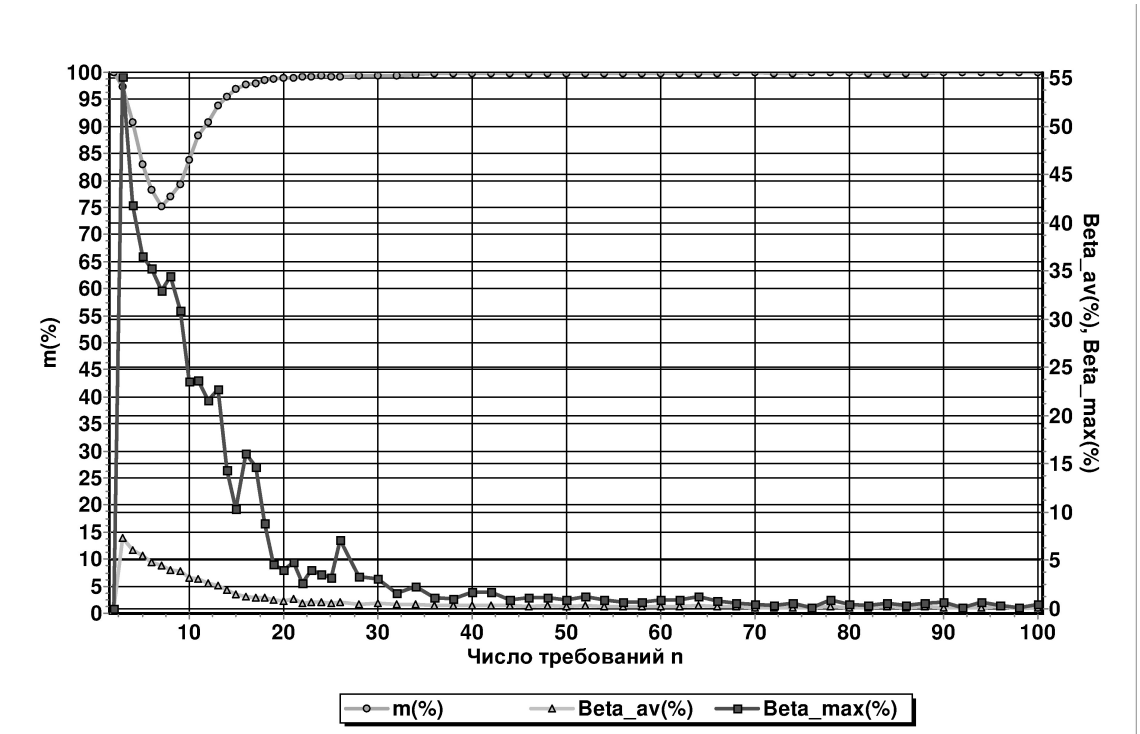

Рис. 1.4: Эффективность алгоритма автора

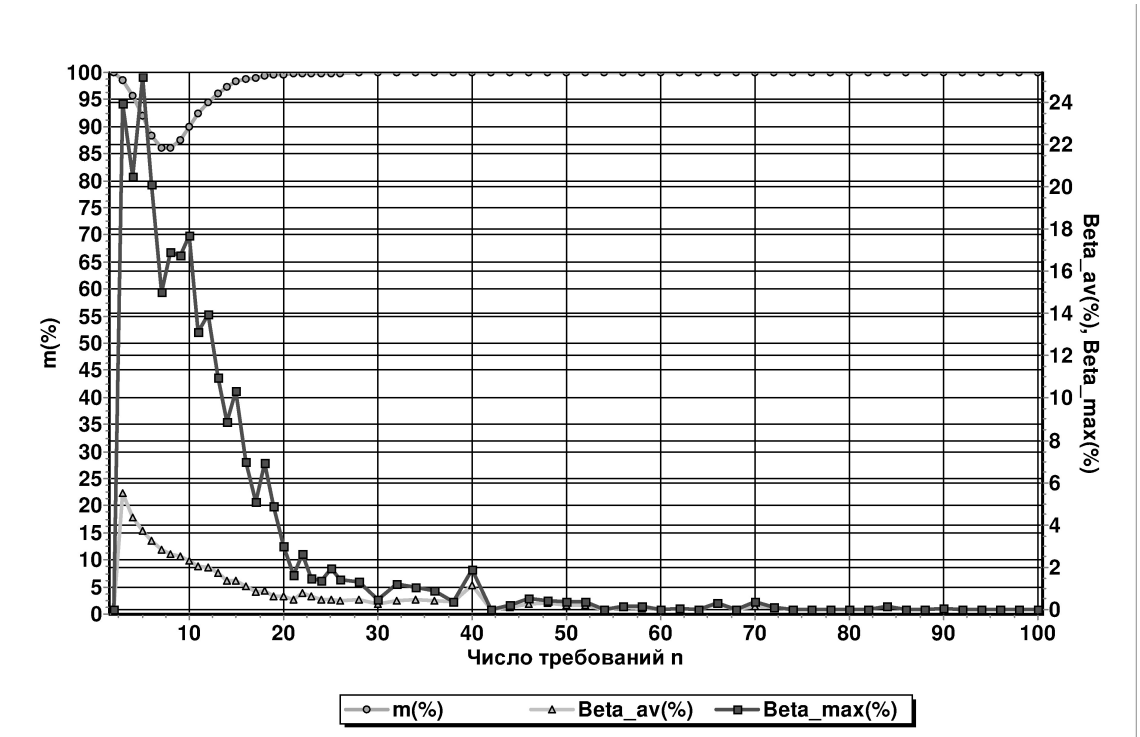

Рис. 1.5: Эффективность алгоритма Хогевена

Графики параметра m имеют схожую зависимость для всех трёх алгоритмов. Сначала параметр m уменьшается с увеличением числа требований. Но затем, начиная с $n = 6,\ 7$ он начинает возрастать и стремится к 100%. Например, для размерности больше 30 алгоритм автора оптимально решает более 99.7% примеров, а алгоритм Хогевена – более 99.9%. Конечно, существуют классы примеров, которые не могут быть решены оптимально этими полиномиальными алгоритмами. Но если брать в расчёт всю совокупность примеров большой размерности, вероятность встретить "плохой" пример очень незначительна.

Если рассмотреть графики параметров β_{av} и $\beta_{\max}$, то можно увидеть, что средняя и максимальная относительная погрешность полученного алгоритмами решения падает с увеличением числа требований. Опять же алгоритм Хогевена показывает наилучшие результаты. Средняя относительная погрешность для этого алгоритма не превышает 1% для размерности $n_1 = 17$ и больше, а максимальная не превысила в наших экспериментах 2% для примеров размерности $n_2 = 23$ и больше. Для алгоритма автора $n_1 = 19$ и $n_2 = 36$, соответственно.

По результатам экспериментально исследования можно выдвинуть гипотезу, что процент "трудных" примеров относительно всего множества примеров задачи $1 \mid r_j \mid L_{\max}$ падает при увеличении размерности n. Этим может быть объяснен, в частности, такой факт, что на практике точные неполиномиальные алгоритмы довольно быстро находят оптимальное решение для примеров большой размерности, несмотря на NP-сложность рассматриваемой задачи, смотрите, например, [101, 142]. В главе 3 мы рассмотрим подробнее эти точные алгоритмы.

1.6 Свойства оптимальных расписаний общего случая задачи $1|r_j|L_{\max}$

Опишем две процедуры разбиения расписания $\pi \in \Pi(N', t')$, $|N'| = n' \neq 0$, на совокупность частичных расписаний.

Пусть $N' \subseteq N$, $N' \neq \emptyset$, $t' \geq t$, $\pi \in \Pi(N', t')$. Будем обозначать:

$r_{\min}(N') = \min\limits_{i \in N'} r_i$ – минимальный момент поступления требований множества N';

$p_{\min}(N') = \min\limits_{i \in N'} p_i$ – минимальная продолжительность обслуживания требований множества N';

$d_{\min}(N') = \min\limits_{i \in N'} d_i$ – минимальный директивный срок требований множества N';

$r_{\max}(N') = \max\limits_{i \in N'} r_i$ – максимальный момент поступления требований множества N';

$p_{\max}(N') = \max\limits_{i \in N'} p_i$ – максимальная продолжительность обслуживания требований множества N';

$d_{\max}(N') = \max\limits_{i \in N'} d_i$ – максимальный директивный срок требований множества N';

$C_{\max}(\pi) = \max\limits_{j \in N'} {}_j(\pi)$ – момент завершения обслуживания всех требований расписания π;

$J^*(\pi) = \{j \in N' : L_{\max}(\pi) = L_j(\pi)\}$ – множество требований из расписания π, на которых достигается значение целевой функции $L_{\max}$;

$J_{d_{\min}}(N') = \{j \in N' : d_j = d_{\min}(N')\}$ – множество требований из N', имеющих минимальный директивный срок;

$\Pi^*(N', t') = \{\pi^* \in \Pi(N', t') : L_{\max}(\pi^*) = \min\limits_{\pi \in \Pi(N', t')} L_{\max}(\pi)\}$ – множество оптимальных расписаний на множестве $\Pi(N', t')$;

$\vec{\Pi}_r(N', t')$ – множество расписаний обслуживания требований множества N' с момента времени t', составленных в порядке неубывания моментов поступления требований. Заметим, что любое расписание $\pi \in \vec{\Pi}_r(N', t')$ является оптимальным по быстродействию расписанием (см. [64], стр. 110);

$\vec{\Pi}_d(N', t')$ – множество расписаний обслуживания требований множества N' с момента времени t', составленное в порядке неубывания директивных сроков завершения обслуживания требований.

Если обслуживание требования i предшествует обслуживанию требования j при расписании π, то это будем обозначать через $i \xrightarrow{\pi} j$. Запись $i \xrightarrow{\pi} N'$ означает, что $i \xrightarrow{\pi} j \ \forall \ j \in N'$, а запись $N' \xrightarrow{\pi} N''$ означает, что $i \xrightarrow{\pi} j \ \forall \ i \in N'$, $j \in N''$.

Алгоритм 1.5

Выберем произвольно для расписания $\pi = (j_1, \ldots, j_{n'})$ требование $j_{d_{\min}} \in J_{d_{\min}}(N')$. Пусть $j_{d_{\min}} = j_i$. Полагаем $N^1 = \emptyset$, если $i = 1$, и $N^1 = \{j_1, \ldots, j_{i-1}\}$, $\pi^1 = (j_1, \ldots, j_{i-1})$, если $1 < i \leq n'$. Полагаем $N^2 = \emptyset$, если $i = n'$, и $N^2 = \{j_{i+1}, \ldots, j_{n'}\}$, $\pi^2 = (j_{i+1}, \ldots, j_{n'})$, если $1 \leq i < n'$. Расписание π представим следующим образом: $\pi = (\pi^1, j_i, \pi^2)$, где $\pi^1 \in \Pi(N^1, t')$, $\pi^2 \in \Pi(N^2, C_{\max}(\pi^1, j_i))$.

Алгоритм 1.6

Выберем произвольно для расписания $\pi = (j_1, \ldots, j_{n'})$ требование $j_d^1 \in J_{d_{\min}}(N')$. Пусть $j_d^1 = j_{i_1}$. Полагаем $N^1 = \emptyset$, если $i_1 = 1$, и $N^1 = \{j_s \in N' : 1 \le s \le i_1 - 1\}$, $\pi^1 = (j_1, \ldots, j_{i_1-1})$, если $i_1 > 1$. Полагаем $N_2 = N' \setminus (N^1 \cup \{j_d^1\})$. Пусть уже известны j_d^{k-1}, N^{k-1}, N_k, π^{k-1}, $k > 1$.

Если $N_k \neq \emptyset$, то выбираем требование $j_d^k \in J_{d_{\min}}(N_k)$. Пусть $j_d^k = j_{i_k}$. Полагаем $N^k = \emptyset$, если $i_k = i_{k-1} + 1$, и $N^k = \{j_s \in N' : i_{k-1} + 1 \le s \le i_k - 1\}$, $\pi^k = (j_{i_{k-1}+1}, j_{i_{k-1}+2}, \ldots, j_{i_k-1})$, если $i_k > i_{k-1} + 1$. Полагаем $N_{k+1} = N_k \setminus (N^k \cup \{j_d^k\})$. Пусть при некотором $k = m$ выполняется $N_m = \emptyset$, тогда расписание π представим в следующем виде: $\pi = (\pi^1, j_d^1, \pi^2, j_d^2, \ldots, \pi^m, j_d^m)$, $\pi^i \in \Pi(N^i, C^i_{\max})$, $i = \overline{1,m}$, $C^1_{\max} = t'$, $C^i_{\max} = C_{\max}(\pi^1, j_d^1, \ldots, \pi^{i-1}, j_d^{i-1})$, $\forall\, i = \overline{2,m}$.

Сформулируем свойства оптимальных расписаний задачи минимизации максимального временно́го смещения. Следующие лемма и следствие дают возможность определить, на каких требованиях может достигаться значение целевой функции при любом расписании $\pi \in \Pi(N', t')$, в том числе и оптимальном.

Лемма 1.5 *Пусть* $N' \subseteq N$, $N' \neq \emptyset$, $t' \ge t$, $j_{d_{\min}} \in J_{d_{\min}}(N')$, $\pi = (\pi^1, j_{d_{\min}}, \pi^2) \in \Pi(N', t')$, *где* N^1, N^2, π^1, π^2 *определены согласно алгоритму 1.5. Тогда существует такое требование* $j^* \in J^*(\pi)$, *что* $j^* \in N^2 \cup \{j_{d_{\min}}\}$.

Доказательство. Из определения $J_{d_{\min}}(N')$ следует, что $d_j \ge d_{j_{d_{\min}}}$ $\forall\, j \in N^1$. Кроме того, $C_j(\pi) \le C_{j_{d_{\min}}}(\pi)$ $\forall\, j \in N^1$. Тогда $L_j(\pi) \le L_{j_{d_{\min}}}(\pi)$ $\forall\, j \in N^1$, и, значит, $\max\limits_{j \in N^1} L_j(\pi) \le L_{j_{d_{\min}}}$.

Поэтому

$$L_{\max}(\pi) = \max\left\{\max_{j \in N^1} L_j(\pi), L_{j_{d_{\min}}}(\pi), \max_{j \in N^2} L_j(\pi)\right\} =$$
$$= \max\left\{L_{j_{d_{\min}}}(\pi), \max_{j \in N^2} L_j(\pi)\right\}.$$

Отсюда следует утверждение леммы. □

Следствие 1.4 *Пусть* $N' \subseteq N$, $N' \neq \emptyset$, $t' \ge t$ *и расписание* $\pi = (\pi^1, j_d^1, \pi^2, j_d^2, ..., \pi^m, j_d^m) \in \Pi(N', t')$, *где* m, j_d^i, N^i, π^i, $i = \overline{1,m}$, *определены согласно алгоритму 1.6. Тогда существует такое требование* $j^* \in J^*(\pi)$, *что* $j^* \in \{j_d^1, j_d^2, ..., j_d^m\}$.

Доказательство. Пусть $\pi = (\pi^1, j_d^1, \pi^2, j_d^2, ..., \pi^m, j_d^m)$ – произвольное расписание множества $\Pi(N', t')$. Из леммы 1.5 следует существование такого

$j^* \in J^*(\pi)$, что $j^* \in N' \setminus N^1$. Для любых $j \in N^k$, $k = \overline{2,m}$, верно $C_j(\pi') \leq C_{j_d^k}(\pi')$. Кроме того, из определения требований $j_d^k \ \forall \ k = \overline{2,m}$, следует, что $d_j \geq d_{j_d^k}$. Поэтому $L_j(\pi) \leq L_{j_d^k}(\pi) \ \forall \ k = 2,...,m, \ j \in N^k$. Отсюда $L_{\max}(\pi) = \max\{L_{j_d^1}(\pi), \max\limits_{k=\overline{2,m}} L_{j_d^k}(\pi)\}$. Следствие доказано. □

В следующих трех утверждениях описывается порядок обслуживания требований в частичных расписаниях, из которых состоит оптимальное расписание. В леммах 1.6, 1.7 доказывается существование оптимального расписания, при котором требования, предшествующие требованию с минимальным директивным сроком, обслуживаются по неубыванию моментов поступления, а требования, следующие за ним, упорядочены оптимальным образом. В теореме 1.5 определяется порядок обслуживания требований в частичных расписаниях π^k, $k = 1,...,m$, определенных в алгоритме 1.6.

Лемма 1.6 *Пусть $N' \subseteq N$, $N' \neq \emptyset$, $t' \geq t$, $j_{d_{\min}} \in J_{d_{\min}}(N')$. Тогда найдутся такие подмножества $N^1, N^2 \subset N'$, $N^1 \cup N^2 = N'$, что расписание $\pi^* = (\vec{\pi}_r^1, j_{d_{\min}}, \pi^2) \in \Pi(N', t')$, при любом $\vec{\pi}_r^1 \in \vec{\Pi}_r(N^1, t')$ и некотором $\pi^2 \in \Pi(N^2, C_{\max}(\vec{\pi}_r^1, j_{d_{\min}}))$, будет оптимальным.*

Доказательство. Пусть $\pi \in \Pi(N', t')$ – оптимальное расписание. Согласно алгоритму 1.5 найдутся такие $N^1, N^2 \subset N'$ и соответствующие им частичные расписания π^1, π^2, что

$$\pi = (\pi^1, j_{d_{\min}}, \pi^2),$$

где $\pi^1 \in \Pi(N^1, t')$, $\pi^2 \in \Pi(N^2, C_{\max}(\pi^1, j_{d_{\min}}))$, $j_{d_{\min}} \in J_{d_{\min}}(N')$.

Построим расписание

$$\pi' = (\vec{\pi}_r^1, j_{d_{\min}}, \pi^2), \ \vec{\pi}_r^1 \in \vec{\Pi}_r(N^1, t'),$$

отличающееся от π порядком обслуживания требований множества N^1. Так как расписание $\vec{\pi}_r^1$ является оптимальным по быстродействию для требований множества N^1 с момента времени t', то

$$C_{\max}(\vec{\pi}_r^1) \leq C_{\max}(\pi^1). \tag{1.40}$$

Из леммы 1.5 следует существование таких требований $j^* \in J^*(\pi)$ и $j'^* \in J^*(\pi')$, что $j^*, j'^* \in N^2 \cup \{j_{d_{\min}}\}$. Поэтому

$$L_{\max}(\pi) - L_{\max}(\pi') = \max_{j \in N^2 \cup \{j_{d_{\min}}\}} L_j(\pi) - \max_{j \in N^2 \cup \{j_{d_{\min}}\}} L_j(\pi'). \tag{1.41}$$

Поскольку порядок обслуживания требований множества $N^2 \cup \{j_{d_{\min}}\}$ не изменился при переходе от расписания π к расписанию π', то согласно (1.40) $C_j(\pi) \geq C_j(\pi')\ \forall\ j \in N^2 \cup \{j_{d_{\min}}\}$, а значит, $L_j(\pi) \geq L_j(\pi')$. Отсюда с учётом (1.41) получим $L_{\max}(\pi) \geq L_{\max}(\pi')$. Таким образом, в результате преобразования произвольно выбранного оптимального расписания π получили оптимальное расписание π', удовлетворяющее утверждению леммы. Лемма доказана. □

Лемма 1.7 *Пусть $N' \subseteq N$, $N' \neq \emptyset$, $t' \geq t$, $j_{d_{\min}} \in J_{d_{\min}}(N')$. Тогда найдутся такие подмножества $N^1, N^2 \subset N'$, $N^1 \cap N^2 = \emptyset$, $N^1 \cup N^2 = N'$, что расписание $\pi^* = (\pi^1, j_{d_{\min}}, \pi^{2*}) \in \Pi(N', t')$, при некотором $\pi^1 \in \Pi(N^1, t')$ и любом $\pi^{2*} \in \Pi^*(N^2, C_{\max}(\pi^1, j_{d_{\min}}))$, будет оптимальным.*

Доказательство. Пусть $\pi \in \Pi(N', t')$ – некоторое оптимальное расписание. Тогда, согласно алгоритму 1.5 найдутся такие $N^1, N^2 \subset N'$ и соответствующие им частичные расписания π^1, π^2, что

$$\pi = (\pi^1, j_{d_{\min}}, \pi^2),$$

где $\pi^1 \in \Pi(N^1, t')$, $\pi^2 \in \Pi(N^2, C_{\max}(\pi^1, j_{d_{\min}}))$, $j_{d_{\min}} \in J_{d_{\min}}(N')$.

Построим расписание

$$\pi' = (\pi^1, j_{d_{\min}}, \pi^{2*}),\ \pi^{2*} \in \Pi^*(N^2, C_{\max}(\pi^1, j_{d_{\min}})),$$

отличающееся от π порядком обслуживания требований множества N^2. Так как порядок обслуживания требований множества $N^1 \cup \{j_{d_{\min}}\}$ не изменился при переходе к расписанию π', то $L_{j_{d_{\min}}}(\pi) = L_{j_{d_{\min}}}(\pi')$. Кроме того, $\max\limits_{i \in N^2} L_i(\pi) \geq \max\limits_{i \in N^2} L_i(\pi')$, поскольку расписание π^{2*} является оптимальным на множестве $\Pi(N^2, C_{\max}(\pi^1, j_{d_{\min}}))$. Значит,

$$\max_{j \in N^2 \cup \{j_{d_{\min}}\}} L_j(\pi) \geq \max_{j \in N^2 \cup \{j_{d_{\min}}\}} L_j(\pi'). \tag{1.42}$$

Из леммы 1.5 следует существование таких требований $j^* \in J^*(\pi)$ и $j'^* \in J^*(\pi')$, что $j^*, j'^* \in N^2 \cup \{j_{d_{\min}}\}$, а значит, справедливо равенство (1.41). Тогда из (1.41), (1.42) следует неравенство $L_{\max}(\pi) \geq L_{\max}(\pi')$. Таким образом, в результате преобразования произвольно выбранного оптимального расписания π получили расписание π' удовлетворяющее утверждению леммы. Лемма доказана. □

Теорема 1.5 *Пусть $N' \subseteq N$, $N' \neq \emptyset$, $t' \geq t$. Существует оптимальное расписание $\pi^* = (\vec{\pi}_r^1, j_d^1, \vec{\pi}_r^2, j_d^2, ..., \vec{\pi}_r^m, j_d^m) \in \Pi(N', t')$, где $\vec{\pi}_r^i \in \vec{\Pi}_r(N^i, C_{\max}^i)$, $i = 1, \ldots, m$, $C_{\max}^1 = t'$, $C_{\max}^i = C_{\max}(\vec{\pi}_r^1, j_d^1, ..., \vec{\pi}_r^{i-1}, j_d^{i-1})$, $i = 2, \ldots, m$, а значение m, требования j_d^i и множества $N^1, ..., N^m \subset N'$ для всех $i = 1, \ldots, m$ определены в алгоритме 1.6.*

Доказательство. Пусть $\pi \in \Pi(N', t')$ – оптимальное расписание. Тогда согласно алгоритму 1.6 найдутся значение m, требования j_d^i, множества $N^i \subset N'$, $i = 1, \ldots, m$, такие, что

$$\pi = (\pi^1, j_d^1, \pi^2, j_d^2, ..., \pi^m, j_d^m),$$

где $\pi^i \in \Pi(N^i, C_{\max}^i)$, $C_{\max}^1 = t'$, $C_{\max}^i = C_{\max}(\pi^1, j_d^1, ..., \pi^{i-1}, j_d^{i-1})$, $i = 2, \ldots, m$.

Построим расписание

$$\pi' = (\vec{\pi}_r^1, j_d^1, \vec{\pi}_r^2, j_d^2, ..., \vec{\pi}_r^m, j_d^m),$$

отличающееся от π порядком обслуживания требований множеств N^1,..., N^m.

Согласно следствию 1.4 существуют такие требования $j^* \in J^*(\pi)$ и $j'^* \in J^*(\pi')$, что $j^*, j'^* \in \{j_d^1, ..., j_d^m\}$. Поэтому

$$L_{\max}(\pi) - L_{\max}(\pi') = \max_{i=\overline{1,m}} L_{j_d^i}(\pi) - \max_{i=\overline{1,m}} L_{j_d^i}(\pi').$$

Поскольку расписания $\vec{\pi}_r^1$,..., $\vec{\pi}_r^m$ являются оптимальными по быстродействию, то $C_{\max}(\pi^i) \geq C_{\max}(\vec{\pi}_r^i)$ $\forall\ i = \overline{1,m}$. Отсюда $C_{j_d^i}(\pi) \geq C_{j_d^i}(\pi')$ и, следовательно, $L_{j_d^i}(\pi) \geq L_{j_d^i}(\pi')$ $\forall\ i = \overline{1,m}$. Поэтому $L_{\max}(\pi) - L_{\max}(\pi') \geq 0$, т.е. в результате преобразования произвольно выбранного оптимального расписания π получили расписание π', удовлетворяющее утверждению теоремы. Теорема доказана. □

Следующая лемма доказывает существование оптимального расписания, в котором требования с моментами поступления, не меньшими чем у требования с минимальным директивным сроком, следуют после него.

Лемма 1.8 *Пусть $N' \subseteq N$, $N' \neq \emptyset$, $t' \geq t$, $j_{d_{\min}} \in J_{d_{\min}}(N')$. Тогда существует оптимальное расписание $\pi^* \in \Pi(N', t')$ такое, что справедливо $j_{d_{\min}} \xrightarrow{\pi^*} j$ для всех $j \in N' \setminus \{j_{d_{\min}}\}$, удовлетворяющих неравенству $r_{j_{d_{\min}}} \leq r_j$.*

Доказательство. Пусть $\pi \in \Pi(N', t')$ – оптимальное расписание. Согласно алгоритму 1.5 найдутся такие подмножества $\tilde{N}, N^2 \subset N'$ и соответствующие им частичные расписания $\tilde{\pi}$, π^2, что π можно представить в виде

$\pi = (\tilde{\pi}, j_{d_{\min}}, \pi^2)$, где $\tilde{\pi} \in \Pi(\tilde{N}, t')$, $\pi^2 \in \Pi(N^2, C_{\max}(\tilde{\pi}, j_{d_{\min}}))$. Выберем требование $j \in N'$ так, чтобы оно было в расписании π ближайшим слева к требованию $j_{d_{\min}}$ и для него выполнялось неравенство $r_{j_{d_{\min}}} \leq r_j$. Отметим, что если слева от $j_{d_{\min}}$ указанного требования j не найдется, то для расписания π будет выполняться утверждение леммы.

Согласно алгоритму 1.5 найдутся такие подмножества $\tilde{N}^1, \tilde{N}^2 \subset \tilde{N}$ и соответствующие им частичные расписания $\tilde{\pi}^1$, $\tilde{\pi}^2$, что $\tilde{\pi} = (\tilde{\pi}^1, j, \tilde{\pi}^2)$, где $\tilde{\pi}^1 \in \Pi(\tilde{N}^1, t')$, $\tilde{\pi}^2 \in \Pi(\tilde{N}^2, C_{\max}(\tilde{\pi}^1, j))$.

Построим расписание

$$\pi = (\tilde{\pi}^1, j, \tilde{\pi}^2, j_{d_{\min}}, \pi^2).$$

Согласно выбору требования j для всех $i \in \tilde{N}^2$ выполняются неравенства $r_i \leq r_{j_{d_{\min}}} \leq r_j$. Так как требование не может обслуживаться ранее своего момента поступления, то $r_j \leq C_j(\pi) - p_j$. Отсюда следуют неравенства

$$r_i \leq r_{j_{d_{\min}}} \leq r_j \leq C_j(\pi) - p_j \ \ \forall \ \ i \in \tilde{N}^2. \tag{1.43}$$

Построим расписание

$$\pi' = (\tilde{\pi}^1, \tilde{\pi}^2, j_{d_{\min}}, j, \pi^2),$$

отличающееся от расписания π порядком обслуживания требования j. Из (1.43) следует, что к моменту времени $(C_j(\pi) - p_j)$ требования множества $\tilde{N}^2 \cup \{j_{d_{\min}}\}$ поступили на обслуживание, т.е. при расписаниях π и π'', где $\pi'' = (\tilde{\pi}^2, j_{d_{\min}}) \in \Pi(\tilde{N}^2 \cup \{j_{d_{\min}}\}, \max\{C_{\max}(\tilde{\pi}^1), r_j\})$, они обслуживаются без простоев прибора. Это означает, что

$$C_{j_{d_{\min}}}(\pi) = \max\{C_{\max}(\tilde{\pi}^1), r_j\} + p_j + \sum_{i \in \tilde{N}^2} p_i + p_{j_{d_{\min}}} \tag{1.44}$$

и

$$C_{j_{d_{\min}}}(\pi'') = \max\{C_{\max}(\tilde{\pi}^1), r_j\} + \sum_{i \in \tilde{N}^2} p_i + p_{j_{d_{\min}}}.$$

Поскольку порядок обслуживания требований множества $\tilde{N}^2 \cup \{j_{d_{\min}}\}$ одинаков при расписаниях π', π'' и $C_{\max}(\tilde{\pi}^1) \leq \max\{C_{\max}(\tilde{\pi}^1), r_j\}$, т.е. момент начала их обслуживания не больше при расписании π', чем при π'', то

$$C_{j_{d_{\min}}}(\pi') \leq \max\{C_{\max}(\tilde{\pi}^1), r_j\} + \sum_{i \in \tilde{N}^2} p_i + p_{j_{d_{\min}}}. \tag{1.45}$$

Тогда с учётом (1.44), (1.45) будем иметь $C_{j_{d_{\min}}}(\pi) - C_{j_{d_{\min}}}(\pi') \geq p_j$ или, что тоже самое

$$C_{j_{d_{\min}}}(\pi) \geq C_{j_{d_{\min}}}(\pi') + p_j. \tag{1.46}$$

Из (1.46) следует

$$L_{j_{d_{\min}}}(\pi) \geq L_{j_{d_{\min}}}(\pi') + p_j. \tag{1.47}$$

Покажем теперь, что

$$C_{j_{d_{\min}}}(\pi) \geq C_j(\pi'). \tag{1.48}$$

Отметим, что $C_j(\pi') = \max\{C_{j_{d_{\min}}}(\pi'), r_j\} + p_j$. Пусть $C_{j_{d_{\min}}}(\pi') \geq r_j$. Тогда с учётом (1.46) $C_{j_{d_{\min}}}(\pi) - C_j(\pi') = C_{j_{d_{\min}}}(\pi) - C_{j_{d_{\min}}}(\pi') - p_j \geq 0$, и неравенство (1.48) в этом случае справедливо. Пусть теперь

$$C_{j_{d_{\min}}}(\pi') < r_j. \tag{1.49}$$

Тогда с учётом (1.44)

$$C_{j_{d_{\min}}}(\pi) - C_j(\pi') = C_{j_{d_{\min}}}(\pi) - r_j - p_j = \max\{C_{\max}(\pi^1), r_j\} + \sum_{i \in \tilde{N}^2} p_i + p_{j_{d_{\min}}} - r_j.$$

Из структуры расписания π' видно, что $C_{j_{d_{\min}}}(\pi') \geq C_{\max}(\tilde{\pi}^1)$. Поэтому, учитывая (1.49), $C_{\max}(\tilde{\pi}^1) < r_j$, и, значит, $C_{j_{d_{\min}}}(\pi) - C_j(\pi') = \sum_{i \in \tilde{N}^2} p_i + p_{j_{d_{\min}}} \geq 0$.

Таким образом, неравенство (1.48) доказано.

Так как $d_{j_{d_{\min}}} \leq d_j$, то из (1.48) следует неравенство

$$L_{j_{d_{\min}}}(\pi) \geq L_j(\pi'). \tag{1.50}$$

Поскольку порядок обслуживания требований множества N^2 не изменился при переходе от расписания π к расписанию π' и справедливо (1.48), то $C_i(\pi) \geq C_i(\pi')\ \forall\ i \in N^2$. Следовательно,

$$L_i(\pi) \geq L_i(\pi')\ \ \forall\ \ i \in N^2. \tag{1.51}$$

По лемме 1.5 существуют такие требования $j^* \in J^*(\pi)$ и $j'^* \in J^*(\pi')$, что $j^* \in N^2 \cup \{j_{d_{\min}}\}$, $j'^* \in N^2 \cup \{j_{d_{\min}}\} \cup \{j\}$. Тогда справедливо

$$\begin{aligned} L_{\max}(\pi) - L_{\max}(\pi') = \max\{L_{j_{d_{\min}}}(\pi), \max_{i \in N^2} L_i(\pi)\} - \\ \max\{L_{j_{d_{\min}}}(\pi'), L_j(\pi'), \max_{i \in N^2} L_i(\pi')\}. \end{aligned} \tag{1.52}$$

Покажем, что

$$L_{\max}(\pi) \geq L_{\max}(\pi'). \tag{1.53}$$

Возможны варианты.

1. Пусть

$$L_{j_{d_{\min}}}(\pi) \geq \max_{i \in N^2} L_i(\pi). \tag{1.54}$$

Тогда с учётом (1.52)

$$L_{\max}(\pi) - L_{\max}(\pi') = L_{j_{d_{\min}}}(\pi) - \max\{L_{j_{d_{\min}}}(\pi'), L_j(\pi'), \max_{i \in N^2} L_i(\pi')\}.$$

Рассмотрим случаи:

a) пусть $\max\{L_{j_{d_{\min}}}(\pi'), L_j(\pi'), \max\limits_{i \in N^2} L_i(\pi')\} = L_{j_{d_{\min}}}(\pi')$. Тогда $L_{\max}(\pi) - L_{\max}(\pi') = L_{j_{d_{\min}}}(\pi) - L_{j_{d_{\min}}}(\pi') \geq p_j \geq 0$, что следует из (1.47);

b) пусть $\max\{L_{j_{d_{\min}}}(\pi'), L_j(\pi'), \max\limits_{i \in N^2} L_i(\pi')\} = L_j(\pi')$. Тогда $L_{\max}(\pi) - L_{\max}(\pi') = L_{j_{d_{\min}}}(\pi) - L_j(\pi') \geq 0$, что следует из (1.50);

c) пусть $\max\{L_{j_{d_{\min}}}(\pi'), L_j(\pi'), \max\limits_{i \in N^2} L_i(\pi')\} = \max\limits_{i \in N^2} L_i(\pi')$. Тогда

$$L_{\max}(\pi) - L_{\max}(\pi') = L_{j_{d_{\min}}}(\pi) - \max_{i \in N^2} L_i(\pi').$$

Поскольку справедливо (1.54) и из (1.51) следует, что $\max\limits_{i \in N^2} L_i(\pi) \geq \max\limits_{i \in N^2} L_i(\pi')$, то $L_{j_{d_{\min}}}(\pi) \geq \max\limits_{i \in N^2} L_i(\pi')$ и, значит, $L_{\max}(\pi) - L_{\max}(\pi') \geq 0$. Таким образом, неравенство (1.53) в этом случае доказано.

2. Пусть

$$\max_{i \in N^2} L_i(\pi) > L_{j_{d_{\min}}}(\pi). \tag{1.55}$$

Тогда с учётом (1.52)

$$L_{\max}(\pi) - L_{\max}(\pi') = \max_{i \in N^2} L_i(\pi) - \max\{L_{j_{d_{\min}}}(\pi'), L_j(\pi'), \max_{i \in N^2} L_i(\pi')\}.$$

По аналогии рассмотрим случаи:

a) пусть $\max\{L_{j_{d_{\min}}}(\pi'), L_j(\pi'), \max\limits_{i \in N^2} L_i(\pi')\} = L_{j_{d_{\min}}}(\pi')$. Тогда

$$L_{\max}(\pi) - L_{\max}(\pi') = \max_{i \in N^2} L_i(\pi) - L_{j_{d_{\min}}}(\pi') \geq \max_{i \in N^2} L_i(\pi) - L_{j_{d_{\min}}}(\pi) + p_j,$$

что следует из (1.47). Поскольку справедливо (1.55), то $L_{\max}(\pi) - L_{\max}(\pi') > p_j \geq 0$;

b) пусть $\max\{L_{j_{d_{\min}}}(\pi'), L_j(\pi'), \max\limits_{i \in N^2} L_i(\pi')\} = L_j(\pi')$. Тогда

$$L_{\max}(\pi) - L_{\max}(\pi') = \max_{i \in N^2} L_i(\pi) - L_j(\pi').$$

Поскольку справедливо (1.55), то $L_{\max}(\pi) - L_{\max}(\pi') > L_{j_{d_{\min}}}(\pi) - L_j(\pi') \geq 0$, последнее неравенство следует из (1.50);

c) пусть $\max\{L_{j_{d_{\min}}}(\pi'), L_j(\pi'), \max\limits_{i \in N^2} L_i(\pi')\} = \max\limits_{i \in N^2} L_i(\pi')$. Тогда $L_{\max}(\pi) - L_{\max}(\pi') = \max\limits_{i \in N^2} L_i(\pi) - \max\limits_{i \in N^2} L_i(\pi') \geq 0$, что следует из (1.51). Неравенство (1.53) доказано.

Таким образом, в результате преобразования произвольно выбранного оптимального расписания π путём перестановки из левой (относительно $j_{d_{\min}}$) части в правую требования j, для которого $r_{j_{d_{\min}}} \geq r_j$, получили оптимальное расписание π', где порядок обслуживания требований $j_{d_{\min}}$ и j удовлетворяет утверждению леммы. Следовательно, любое оптимальное расписание в результате таких последовательных перестановок можно привести к оптимальному расписанию, удовлетворяющему утверждению леммы. Лемма доказана. □

Далее сформулированы свойства оптимальных расписаний, касающиеся порядка обслуживания требований в частичных расписаниях, из которых состоит оптимальное, при добавлении определенных требований к исходному множеству.

Пусть $N' \subset N$ и в множестве N существуют требования j' и j'' такие, что

$$r_{j'} \leq r_j, \;\; d_{j'} \geq d_j \;\; \forall \;\; j \in N', \tag{1.56}$$

$$r_{j''} \geq r_j, \;\; d_{j''} \leq d_j \;\; \forall \;\; j \in N'. \tag{1.57}$$

Индекс j' будем обозначать через $j^0(N')$, а индекс j'' – через $j^{n+1}(N')$. Если в множестве N требований j' и j'', удовлетворяющих условиям (1.56), (1.57) нет, то определяем фиктивные требования j' и j'' и параметры $r_{j'}$, $r_{j''}$, $d_{j'}$, $d_{j''}$ так, чтобы выполнялись неравенства (1.56), (1.57), и обозначаем их как и выше через $j^0(N')$ и $j^{n+1}(N')$, соответственно.

Лемма 1.9 *Пусть $\bar{N} \subseteq N$, $t' \geq t$, $N' = \bar{N} \cup \{j^{n+1}(\bar{N})\}$, где $j^{n+1}(\bar{N})$ выбрано согласно (1.57). Тогда найдутся такие подмножества $N^1, N^2 \subset N'$, что $N^1 \cup N^2 = N' \setminus \{j^{n+1}(\bar{N})\}$, и расписание $\pi^* = (\vec{\pi}_r^1, j^{n+1}(\bar{N}), \vec{\pi}_d^2) \in \Pi(N', t')$ при любых $\vec{\pi}_r^1 \in \vec{\Pi}_r(N^1, t')$ и $\vec{\pi}_d^2 \in \vec{\Pi}_d(N^2, C_{\max}(\vec{\pi}_r^1, j^{n+1}(\bar{N})))$ будет оптимальным.*

Доказательство. В силу (1.57) имеем $j^{n+1}(\bar{N}) \in J_{d_{\min}}(N')$. Согласно леммам 1.6, 1.7 выберем подмножества $N^1, N^2 \subset N' \subset N \cup \{j^{n+1}(\bar{N})\}$ так, чтобы $N^1 \cup N^2 = \bar{N}$ и расписание $\pi = (\vec{\pi}_r^1, j^{n+1}(\bar{N}), \pi^{2*})$, где $\vec{\pi}_r^1 \in \vec{\Pi}_r(N^1, t')$, $\pi^{2*} \in \Pi^*(N^2, C_{\max}(\vec{\pi}_r^1, j^{n+1}(\bar{N})))$, было оптимальным.

Из (1.57) следует, что $r_{j^{n+1}(\bar{N})} \geq r_j$, $j \in N^2$, значит, к моменту начала обслуживания требования $j^{n+1}(\bar{N})$ все требования множества N^2 уже поступили на обслуживание. Следовательно, расписание, составленное в порядке неубывания директивных сроков $\vec{\pi}_d^2 \in \vec{\Pi}_d(N^2, C_{\max}(\vec{\pi}_r^1, j^{n+1}(\bar{N})))$, будет оптимальным ([64], стр. 110) для требований множества N^2 с момента времени $C_{\max}(\vec{\pi}_r^1, j^{n+1}(\bar{N}))$. Поэтому $L_{\max}(\pi^{2*}) = L_{\max}(\vec{\pi}_d^2)$, а значит, $(\vec{\pi}_r^1, j^{n+1}(\bar{N}), \vec{\pi}_d^2)$ – оптимальное расписание. Лемма доказана. □

Таким образом, существует оптимальное расписание обслуживания произвольного множества требований N при котором требования **до** требования $j^{n+1}(N)$ обслуживаются по неубыванию моментов поступления, а **после** требования $j^{n+1}(N)$ – по неубыванию директивных сроков. Остаётся открытым вопрос: на какой позиции должно обслуживаться требование $j^{n+1}(N)$ и какие требования должны предшествовать и последовать этому требованию.

Лемма 1.10 *Пусть* $\bar{N} \subseteq N$, $t' \geq t$, $N' = \bar{N} \cup \{j^0(\bar{N}), j^{n+1}(\bar{N})\}$, *где* $j^0(\bar{N})$, $j^{n+1}(\bar{N})$ *выбраны согласно (1.56), (1.57). Тогда существует оптимальное расписание* $\pi^* \in \Pi(N', t')$, *при котором справедливо либо* $j^0(\bar{N}) \xrightarrow{\pi^*} j$, *либо* $j \xrightarrow{\pi^*} j^0(\bar{N})$ *для каждого* $j \in N' \setminus \{j^0(\bar{N})\}$.

Доказательство. Согласно лемме 1.9 выберем подмножества $N^1, N^2 \subset N' \subset N \cup \{j^0(\bar{N}), j^{n+1}(\bar{N})\}$ так, чтобы $N^1 \cup N^2 = N' \setminus \{j^{n+1}(\bar{N})\}$ и расписание

$$\pi = (\vec{\pi}_r^1, j^{n+1}(\bar{N}), \vec{\pi}_d^2) \in \Pi(N', t'),$$

где $\vec{\pi}_r^1 \in \vec{\Pi}_r(N^1, t')$, $\vec{\pi}_d^2 \in \vec{\Pi}_d(N^2, C_{\max}(\vec{\pi}_r^1, j^{n+1}(\bar{N})))$, было оптимальным.

Пусть $j^0(\bar{N}) \in N^1$. Из (1.56) следует, что $r_{j^0(\bar{N})} \leq r_j \ \forall \ j \in N^1$. Тогда существует расписание $\bar{\pi}^1 \in \vec{\Pi}_r(N^1, t')$, при котором требование $j^0(\bar{N})$ обслуживается первым по порядку, т.е. $j^0(\bar{N}) \xrightarrow{\bar{\pi}^1} j \ \forall \ j \in N^1$. Построим расписание

$$\pi' = (\bar{\pi}^1, j^{n+1}(\bar{N}), \vec{\pi}_d^2),$$

которое отличается от π порядком обслуживания требований множества N^1 и при котором выполняется $j^0(\bar{N}) \xrightarrow{\pi'} j \ \forall \ j \in N' \setminus \{j^0(\bar{N})\}$. Покажем, что π' – оптимальное расписание. Из (1.57) следует, что $d_{j^{n+1}(\bar{N})} \leq d_j \ \forall \ j \in N' \setminus \{j^{n+1}(\bar{N})\}$, поэтому $j^{n+1}(\bar{N}) \in J_{d_{\min}}(N')$. Тогда из леммы 1.5 следует существование таких требований $j^* \in J^*(\pi)$ и $j'^* \in J^*(\pi')$, что $j^*, j'^* \in N^2 \cup \{j^{n+1}(\bar{N})\}$, а значит,

$$L_{\max}(\pi) = \max_{j \in N^2 \cup \{j^{n+1}(\bar{N})\}} L_j(\pi) \tag{1.58}$$

и

$$L_{\max}(\pi') = \max_{j \in N^2 \cup \{j^{n+1}(\bar{N})\}} L_j(\pi'). \qquad (1.59)$$

Поскольку $\vec{\pi}_r^1, \bar{\pi}^1 \in \vec{\Pi}_r(N^1, t')$, то $C_{\max}(\vec{\pi}_r^1) = C_{\max}(\bar{\pi}^1)$. Кроме того, порядок обслуживания требований множества $N^2 \cup \{j^{n+1}(\bar{N})\}$ не изменился при переходе от расписания π к π'. Отсюда с учётом (1.58), (1.59) $L_{\max}(\pi) = L_{\max}(\pi')$, следовательно, расписание π' является оптимальным.

Пусть $j^0(\bar{N}) \in N^2$. В силу (1.56) $d_{j^0(\bar{N})} \geq d_j \ \forall \ j \in N^2$, тогда существует расписание $\bar{\pi}^2 \in \vec{\Pi}_d(N^2, C_{\max}(\vec{\pi}_r^1, j^{n+1}(\bar{N})))$, при котором требование $j^0(\bar{N})$ обслуживается последним по порядку, т.е. $j \overset{\bar{\pi}^2}{\to} j^0(\bar{N}) \ \forall \ j \in N^2$. Построим расписание

$$\pi'' = (\vec{\pi}_r^1, j^{n+1}(\bar{N}), \bar{\pi}^2),$$

которое отличается от π порядком обслуживания требований множества N^2 и при котором выполняется $j \overset{\pi''}{\to} j^0(\bar{N}) \ \forall \ j \in N' \setminus \{j^0(\bar{N})\}$. Покажем, что π'' – оптимальное расписание. Из (1.57) следует, что $d_{j^{n+1}(\bar{N})} \leq d_j \ \forall \ j \in N' \setminus \{j^{n+1}(\bar{N})\}$, поэтому $j^{n+1}(\bar{N}) \in J_{d_{\min}}(N')$. Тогда из леммы 1.5 следует существование таких требований $j^* \in J^*(\pi)$ и $j'^* \in J^*(\pi'')$, что $j^*, j'^* \in N^2 \cup \{j^{n+1}(\bar{N})\}$, а значит, справедливы (1.58), (1.59). Так как $\vec{\pi}_d^2, \bar{\pi}^2 \in \vec{\Pi}_d(N^2, C_{\max}(\vec{\pi}_r^1, j^{n+1}(\bar{N})))$, то расписания $\vec{\pi}_d^2, \bar{\pi}^2$ (в π' и π'' соответственно), составленные в порядке неубывания директивных сроков, будут оптимальными ([64], стр. 110) для требований множества N^2 с момента времени $C_{\max}(\vec{\pi}_r^1, j^{n+1}(\bar{N}))$. Поэтому $L_{\max}(\vec{\pi}_d^2) = L_{\max}(\bar{\pi}^2)$. Кроме того, $C_j(\pi) = C_j(\pi'') \ \forall \ j \in N^1 \cup \{j^{n+1}(\bar{N})\}$. Отсюда с учётом (1.58), (1.59) $L_{\max}(\pi) = L_{\max}(\pi'')$, следовательно, расписание π'' является оптимальным.

Таким образом, в результате преобразования произвольно выбранного оптимального расписания π получили оптимальные расписания π', π'' удовлетворяющие утверждению леммы. Лемма доказана. □

Пусть $N' \subseteq N$, $N' \neq \emptyset$, $t' \geq t$, $j_{d_{\min}} \in J_{d_{\min}}(N')$. Будем считать, что существует процедура построения множества

$$N^* = \{j \in N' \setminus \{j_{d_{\min}}\} : j_{d_{\min}} \overset{\pi^* \in \Pi^*(N', t')}{\to} j, r_j < r_{j_{d_{\min}}}\},$$

требования из которого будут предшествовать требованию $j_{d_{\min}}$ в оптимальном расписании.

Приведём схему построения оптимального расписания на множестве всех расписаний $\Pi(N', t')$.

Алгоритм 1.7

Первоначально полагаем $t_1 = \max\{r_{\min}(N'), t'\}$, $N_1 = N'$. Произвольно выбираем $j_d^1 \in J_{d_{\min}}(N_1)$. Полагаем

$N_1^* = \{j \in N_1 \setminus \{j_d^1\} : j_d^1 \overset{\pi^* \in \Pi^*(N_1,t_1)}{\to} j, r_j < r_{j_d^1}\}$,

$N_2 = (\{j \in N_1 : r_j \geq r_{j_d^1}\} \setminus \{j_d^1\}) \cup N_1^*$,

$N^1 = N_1 \setminus \{N_2 \cup \{j_d^1\}\}$, $\pi_1 = (\vec{\pi}_r^1, j_d^1)$, где $\vec{\pi}_r^1 \in \vec{\Pi}_r(N^1, t_1)$, $t_2 = C_{\max}(\pi_1)$.

Пусть уже известны j_d^{k-1}, N^{k-1}, π_{k-1}, N_k, t_k при $k > 1$.

Если $N_k \neq \emptyset$, то выбираем $j_d^k \in J_{d_{\min}}(N_k)$, и полагаем

$N_k^* = \{j \in N_k \setminus \{j_d^k\} : j_d^k \overset{\pi^* \in \Pi^*(N_k,t_k)}{\to} j, r_j < r_{j_d^k}\}$,

$N_{k+1} = (\{j \in N_k : r_j \geq r_{j_d^k}\} \setminus \{j_d^k\}) \cup N_k^*$,

$N^k = N_k \setminus \{N_{k+1} \cup \{j_d^k\}\}$, $\pi_k = (\pi_{k-1}, \vec{\pi}_r^k, j_d^k)$,

где $\vec{\pi}_r^k \in \vec{\Pi}_r(N^k, C_{\max}(\pi_{k-1}))$, $t_{k+1} = C_{\max}(\pi_k)$.

В противном случае $\pi_{k-1} \in \Pi^*(N', t')$, и процесс заканчивается.

Лемма 1.11 *Пусть $N' \subseteq N$, $N' \neq \emptyset$, $t' \geq t$, $\pi^* \in \Pi^*(N', t')$ такое, что $N^1 \overset{\pi^*}{\to} j_{d_{\min}} \overset{\pi^*}{\to} N^2$, где $N^1, N^2 \subset N'$, $j_{d_{\min}} \in J_{d_{\min}}(N')$. Тогда расписание $(\vec{\pi}_r^1, j_{d_{\min}}, \pi^{2*}) \in \Pi(N', t')$, где $\vec{\pi}_r^1 \in \vec{\Pi}_r(N^1, t')$, $\pi^{2*} \in \Pi^*(N^2, C_{\max}(\vec{\pi}_r^1, j_{d_{\min}}))$, будет оптимальным.*

Доказательство. Пусть $\pi \in \Pi(N', t')$ – оптимальное расписание, при котором справедливо $N^1 \overset{\pi}{\to} j_{d_{\min}} \overset{\pi}{\to} N^2$, где $N^1, N^2 \subset N'$, $j_{d_{\min}} \in J_{d_{\min}}(N')$. Согласно алгоритму 1.5 расписание π можно представить в виде

$$\pi = (\pi^1, j_{d_{\min}}, \pi^2),$$

где $\pi^1 \in \Pi(N^1, t')$, $\pi^2 \in \Pi(N^2, C_{\max}(\pi^1, j_{d_{\min}}))$.

Построим расписание

$$\pi' = (\vec{\pi}_r^1, j_{d_{\min}}, \pi^2),\ \vec{\pi}_r^1 \in \vec{\Pi}_r(N^1, t'),$$

отличающееся от π порядком обслуживания требований множества N^1. Покажем, что

$$L_{\max}(\pi) \geq L_{\max}(\pi'). \tag{1.60}$$

Так как расписание $\vec{\pi}_r^1$ является оптимальным по быстродействию для требований множества N^1 с момента времени t', то справедливо (1.40). Из леммы 1.5 следует существование таких требований $j^* \in J^*(\pi)$ и $j'^* \in J^*(\pi')$, что $j^*, j'^* \in N^2 \cup \{j_{d_{\min}}\}$. Поэтому справедливо (1.41). Поскольку порядок обслуживания требований множества $N^2 \cup \{j_{d_{\min}}\}$ не изменился при переходе от расписания π к расписанию π', то согласно (1.40) имеем $C_j(\pi) \geq C_j(\pi')$ $\forall\ j \in N^2 \cup \{j_{d_{\min}}\}$, а значит, $L_j(\pi) \geq L_j(\pi')$. Отсюда с учётом (1.41) следует справедливость (1.60).

Построим расписание

$$\pi^{''} = (\vec{\pi}_r^1, j_{d_{\min}}, \pi^{2*}),\ \pi^{2*} \in \Pi^*(N^2, C_{\max}(\vec{\pi}_r^1, j_{d_{\min}})),$$

отличающееся от π' порядком обслуживания требований множества N^2. Покажем, что

$$L_{\max}(\pi') \geq L_{\max}(\pi^{''}). \tag{1.61}$$

Так как порядок обслуживания требований множества $N^1 \cup \{j_{d_{\min}}\}$ не изменился при переходе от расписания π' к $\pi^{''}$, то $L_{j_{d_{\min}}}(\pi') = L_{j_{d_{\min}}}(\pi^{''})$. Кроме того, $\max\limits_{j \in N^2} L_j(\pi') \geq \max\limits_{j \in N^2} L_j(\pi^{''})$, поскольку расписание π^{2*} является оптимальным на множестве $\Pi(N^2, C_{j_{d_{\min}}}(\pi))$. Значит,

$$\max_{j \in N^2 \cup \{j_{d_{\min}}\}} L_j(\pi') \geq \max_{j \in N^2 \cup \{j_{d_{\min}}\}} L_j(\pi^{''}). \tag{1.62}$$

Из леммы 1.5 следует существование таких требований $j^{'*} \in J^*(\pi')$ и $j^{''*} \in J^*(\pi^{''})$, что $j^{'*}, j^{''*} \in N^2 \cup \{j_{d_{\min}}\}$, а значит, справедливо равенство

$$L_{\max}(\pi') - L_{\max}(\pi^{''}) = \max_{j \in N^2 \cup \{j_{d_{\min}}\}} L_j(\pi') - \max_{j \in N^2 \cup \{j_{d_{\min}}\}} L_j(\pi^{''}).$$

Отсюда и из (1.62) следует неравенство (1.61).

Таким образом, в результате последовательных преобразований произвольно выбранного оптимального расписания π, при котором справедливо $N^1 \xrightarrow{\pi} j_{d_{\min}} \xrightarrow{\pi} N^2$, получили расписание $\pi^{''}$ удовлетворяющее утверждению леммы. Лемма доказана. □

Теорема 1.6 *Пусть известна процедура отыскания подмножества N^* для любых $N' \subseteq N$, $N' \neq \emptyset$, и $t' \geq t$. Тогда существует алгоритм построения оптимального расписания на множестве $\Pi(N, t)$ трудоёмкости $O(n^2 + nx)$ операций, где $O(x)$ – трудоёмкость процедуры отыскания подмножества N^*.*

Доказательство. Пусть существует процедура отыскания подмножества N^* для любого множества $N' \subseteq N$. Напомним, множество N^* – это множество требований, которые следуют за требованием $j_{d_{\min}}$ при оптимальном расписании. Пусть в алгоритме 1.7 $k \geq 1$. Поскольку $j_d^k \in J_{d_{\min}}(N_k)$, то согласно лемме 1.8 существует оптимальное расписание $\pi^* \in \Pi(N_k, t_k)$ такое, что $j_d^k \xrightarrow{\pi^*} \{j \in N_k : r_j \geq r_{j_d^k}\} \setminus \{j_d^k\}$, и, кроме того, $j_d^k \xrightarrow{\pi^*} N^*$. Следовательно, $N^k \xrightarrow{\pi^*} j_d^k \xrightarrow{\pi^*} N_{k+1}$. Тогда из леммы 1.11 следует существование оптимального расписания $\pi \in \Pi(N_k, t_k)$ такого, что $\pi = (\vec{\pi}_r^k, j_d^k, \pi^{2*})$, где

$\vec{\pi}_r^k \in \vec{\Pi}_r(N^k, t_k)$, $\pi^{2*} \in \Pi(N_{k+1}, C_{\max}(\vec{\pi}_r^k, j_d^k))$. Поэтому на рассматриваемой итерации схемы будет построено начало $(\vec{\pi}_r^k, j_d^k)$ оптимального расписания на множестве $\Pi(N_k, t_k)$, и задача построения оптимального расписания на этом множестве сводится к задаче построения оптимального расписания на $\Pi(N_{k+1}, C_{\max}(\vec{\pi}_r^k, j_d^k))$, которое на следующей итерации схемы будет строится аналогично π. Следовательно, на некотором m-м шаге алгоритма 1.7 таком, что $m \leq n$ и $N_{m-1} = \emptyset$, будет построено оптимальное расписание $\pi_m = (\vec{\pi}_r^1, j_d^1, ..., \vec{\pi}_r^m, j_d^m) \in \Pi^*(N, t)$.

Оценим трудоёмкость алгоритма 1.7. Пусть $O(x)$ – трудоёмкость отыскания N^*. Количество итераций алгоритма 1.7, очевидно, не превосходит n. Оценим трудоёмкость одной итерации. Если перенумеровать требования по неубыванию моментов поступления, то для отыскания j_d^k, N_{k+1} и π_k потребуется $O(n)$ операций. Для нахождения N^k и t_{k+1} также необходимо $O(n)$ операций. И, наконец, трудоёмкость построения N^* составляет $O(x)$ операций. Следовательно, для выполнения одной итерации схемы потребуется $O(n+x)$ операций. Поскольку количество итераций не превышает n и для перенумерации требований необходимо $O(n \log n)$ операций [11], то трудоёмкость построения оптимального расписания составит $O(n^2 + nx)$ операций. Теорема доказана. □

Таким образом, задача построения оптимального расписания сводится к задаче отыскания подмножества N^*. Отметим, что в случае, когда требования можно перенумеровать одновременно по неубыванию моментов поступления и по неубыванию директивных сроков, то по алгоритму 1.7 оптимальное расписание строится за $O(n \log n)$ операций, что совпадает с трудоёмкостью известных алгоритмов решения указанного частного случая [133, 142] задачи. Заметим, что к указанному случаю относятся также случаи одновременно́го поступления требований и одинаковых директивных сроков. Действительно, если первоначально перенумеровать требования по неубыванию директивных сроков и моментов поступления, то на каждой итерации схемы в качестве требования j_d^k можно выбирать требование с минимальным номером, для чего потребуется одна операция. Следовательно, алгоритм 1.7 будет состоять из n итераций. Тогда на k-й итерации $N^* = \emptyset$, $N_{k+1} = N_k \setminus \{j_d^k\}$, $N^k = \emptyset$, $\pi_k = (\pi_{k-1}, j_d^k)$. Поскольку для перенумерации требований потребуется $O(n \log n)$ операций, то трудоёмкость алгоритма в этом случае составит $O(n \log n)$ операций.

Для частных случаев задачи минимизации максимального временно́го смещения алгоритм 1.7 будет состоять из одной итерации и её трудоёмкость соответствует трудоёмкости отыскания множества N^*.

1.7 Свойства оптимальных расписаний частных случаев задачи $1|r_j|L_{\max}$

Пусть на параметры требований множества N накладываются следующие ограничения:

$$r_i \le r_j \;\Rightarrow\; d_i \ge d_j \;\;\forall\;\; i,j \in N. \tag{1.63}$$

Кроме того, к условиям (1.63) могут добавляться ограничения

$$d_j - p_j \le d_{\min}(N) \;\;\forall\;\; j \in N, \tag{1.64}$$

или

$$r_i \le r_j \;\Rightarrow\; d_i - p_i \ge d_j \;\;\forall\;\; i,j \in N,\;\; i \ne j. \tag{1.65}$$

В данном параграфе сформулированы свойства оптимальных расписаний задачи минимизации максимального временно́го смещения при ограничениях (1.63), а также (1.63), (1.64) и (1.63), (1.65). На основании этих свойств в главе 2 построены и обоснованы алгоритмы решения задачи.

Необходимо заметить, что случай (1.63) задачи $1|r_j|L_{\max}$ является NP-трудным [168], полиномиальное сведение приведено к NP-полной задаче Partition (Разбиение).

Лемма 1.12 *Пусть $N' \subseteq N$, $N' \ne \emptyset$, $n' = |N'|$, $t' \ge t$, требование $i \in N'$ таково, что $r_i = r_{\min}(N')$ и, кроме того, выполняются* (1.63). *Тогда существует оптимальное расписание $\pi^* \in \Pi(N', t')$, при котором справедливо либо $i \xrightarrow{\pi^*} j$, либо $j \xrightarrow{\pi^*} i$ для всех $j \in N' \setminus \{i\}$.*

Доказательство. Поскольку $r_i = r_{\min}(N')$, то согласно (1.63), $d_i = d_{\max}(N')$. Отсюда и из (1.56) следует, что $i = j^0(N')$. Так как справедливы (1.63), то найдётся такое требование $l \in N'$, что $r_l = r_{\max}(N')$ и $d_l = d_{\min}(N')$. Тогда из (1.57) следует, что $l = j^{n+1}(N')$.

Если $i = l$, то либо $|N'| = 1$ и тогда утверждение леммы справедливо, либо в силу (1.56), (1.57) $r_i = r_j$, $d_i = d_j$ $\forall\; i,j \in N'$ и, следовательно, любое расписание из множества $\Pi(N', t')$ будет оптимальным, в том числе и удовлетворяющее утверждению леммы.

Пусть $i \ne l$. Поскольку выбраны требования $j^0(N')$ и $j^{n+1}(N')$, тогда из леммы 1.10 следует, что существует оптимальное расписание $\pi^* \in \Pi(N', t')$, при котором справедливо $i \xrightarrow{\pi^*} j$ либо $j \xrightarrow{\pi^*} i$ для каждого $j \in N' \setminus \{i\}$. Лемма доказана. □

Лемма 1.13 *Пусть* $N' \subseteq N$, $N' \neq \emptyset$, $t' \geq t$, *и выполняются* (1.63), (1.64). *Тогда найдется такое* $i \in N'$, *что расписание* $(\vec{\pi}_r, i)$, *где* $\vec{\pi}_r \in \vec{\Pi}_r(N' \setminus \{i\}, t')$, *будет оптимальным, причем* $L_{\max}(\pi^*) = L_i(\pi^*)$.

Доказательство. Поскольку справедливо (1.63), то найдется такое требование $i \in N'$, что $r_i = r_{\max}(N')$ и $d_i = d_{\min}(N')$. Тогда из (1.57) следует, что $i = j^{n+1}(N')$. Согласно лемме 1.9 найдутся такие множества $N^1, N^2 \subset N'$, что $N^1 \cup N^2 = N' \setminus \{i\}$ и расписание $\pi = (\vec{\pi}_r^1, i, \vec{\pi}_d^2)$, где $\vec{\pi}_r^1 \in \vec{\Pi}_r(N^1, t')$, $\vec{\pi}_d^2 \in \vec{\Pi}_d(N^2, C_{\max}(\vec{\pi}_r^1, i))$, будет оптимальным. Если $N^2 = \emptyset$, то утверждение леммы справедливо.

Пусть $N^2 \neq \emptyset$. Пусть требование i обслуживается при расписании π k-м по порядку. Тогда $\vec{\pi}_r^1 = (j_1, ..., j_{k-1})$, $\vec{\pi}_d^1 = (j_{k+1}, ..., j_{n'})$, где $n' = |N'|$, и

$$\pi = (j_1, ..., j_{k-1}, j_k, j_{k+1}, ..., j_{n'}),$$

где $j_k = i$. В силу (1.63) имеем $i \in J_{d_{\min}}(N')$. Тогда согласно лемме 1.5 существует такое $j^* \in J^*(N')$, что $j^* \in N^2 \cup \{j_k\}$ и, следовательно,

$$L_{\max}(\pi) = \max_{j \in N^2 \cup \{j_k\}} L_j(\pi). \tag{1.66}$$

Покажем, что

$$L_{\max}(\pi) = L_{j_{n'}}(\pi). \tag{1.67}$$

Поскольку выполняется (1.63), то $r_{j_k} \geq r_j$ для всех $j \in N^2$ и, следовательно, требования j_k, j_{k+1},..., $j_{n'}$ при расписании π обслуживаются без простоев прибора. Это означает, что выполняются равенства

$$C_{j_{l+1}}(\pi) = C_{j_l}(\pi) + p_{l_{l+1}}, \;\; l = \overline{k, n'-1}. \tag{1.68}$$

Согласно (1.63) $d_{j_k} \leq d_j \;\forall\; j \in N^2$. Кроме того, $\vec{\pi}_d^2 \in \vec{\Pi}_d(N^2, C_{\max}(\vec{\pi}_r^1, j_k))$. Следовательно, расписание $(j_k, \vec{\pi}_d^2)$ составлено в порядке неубывания директивных сроков и $d_{j_l} \leq d_{j_{l+1}} \;\forall\; l = \overline{k, n'-1}$. Тогда в силу ограничений (1.63), (1.64) $d_{j_{l+1}} - p_{j_{l+1}} \leq d_{\min}(N) \leq d_{\min}(N') \leq d_{j_l} \;\forall\; l = \overline{k, n'-1}$. Поэтому с учётом (1.68) $L_{j_l}(\pi) = C_{j_l}(\pi) - d_{j_l} \leq C_{j_l}(\pi) + p_{j_{l+1}} - d_{j_{l+1}} = C_{j_{l+1}}(\pi) - d_{j_{l+1}} = L_{j_{l+1}}(\pi)$ $\forall\; l = \overline{k, n'-1}$. Отсюда и из (1.66) следует (1.67).

Преобразуем расписание π в расписание

$$\pi' = (\vec{\pi}_r, j_{n'}),\ \vec{\pi}_r \in \vec{\Pi}_r(N' \setminus \{j_{n'}\}, t'),$$

отличающееся от расписания π порядком обслуживания требований множества $N' \setminus \{j_{n'}\}$. Покажем теперь, что

$$L_{\max}(\pi') = L_{j_{n'}}(\pi'). \tag{1.69}$$

Пусть $\vec{\pi}_r = (i_1, i_2, ..., i_{n'-1}) \in \vec{\Pi}_r(N' \setminus \{j_{n'}\}, t')$. Тогда из (1.63) следует, что $i_{n'-1} \in J_{d_{\min}}(N' \setminus \{j_{n'}\})$. Отсюда согласно лемме 1.5 существует такое $j^* \in J^*(N' \setminus \{j_{n'}\})$, что $j^* = i_{n'-1}$. Значит

$$L_{\max}(\pi') = \max\{L_{i_{n'-1}}(\pi'), L_{j_{n'}}(\pi')\}. \tag{1.70}$$

Если $d_{i_{n'-1}} \geq d_{j_{n'}}$, то, как видно из структуры расписания π', $C_{i_{n'-1}}(\pi') \leq C_{j_{n'}}(\pi')$. Поэтому $L_{i_{n'-1}}(\pi') \leq L_{j_{n'}}(\pi')$ и равенство (1.69) справедливо. Пусть $d_{i_{n'-1}} < d_{j_{n'}}$. Тогда из (1.63), следует, что $r_{i_{n'-1}} \geq r_{j_{n'}}$, а это означает, что при расписании π' требования $i_{n'-1}$ и $j_{n'}$ обслуживаются без простоев прибора, т.е.

$$C_{j_{n'}}(\pi') = C_{i_{n'-1}}(\pi') + p_{j_{n'}}. \tag{1.71}$$

Из (1.63), (1.64) $d_{j_{n'}} - p_{j_{n'}} \leq d_{\min}(N) \leq d_{\min}(N') \leq d_{i_{n'-1}}$. Тогда с учётом (1.71) $L_{i_{n'-1}}(\pi') = C_{i_{n'-1}}(\pi') - d_{i_{n'-1}} \leq C_{i_{n'-1}}(\pi') + p_{j_{n'}} - d_{j_{n'}} = {}_{j_{n'}} - d_{j_{n'}} = L_{j_{n'}}(\pi')$. Отсюда и из (1.70) следует (1.69).

Так как расписание $\vec{\pi}_r$ составлено в порядке неубывания моментов поступления требований множества $N' \setminus \{j_{n'}\}$, то $C_{\max}(\vec{\pi}^1_r, j_k, j_{k+1}, ..., j_{n'-1}) \geq C_{\max}(\vec{\pi}_r)$ и, значит $C_{j_{n'}}(\pi) \geq C_{j_{n'}}(\pi')$. Отсюда $L_{j_{n'}}(\pi) \geq L_{j_{n'}}(\pi')$ и, учитывая (1.67), (1.69), получим $L_{\max}(\pi) \geq L_{\max}(\pi')$. Таким образом, в результате преобразования оптимального расписания π получили оптимальное расписание π' удовлетворяющее утверждению леммы. Лемма доказана. □

Лемма 1.14 *Пусть $N' \subseteq N$, $N' \neq \emptyset$, $t' \geq t$, и выполняются (1.63), (1.65). Тогда расписание $\pi^* \in \vec{\Pi}_d(N', t')$ будет оптимальным, причём $L_{\max}(\pi^*) = L_{j_{d_{\min}}}(\pi^*)$, $j_{d_{\min}} \in J_{d_{\min}}(N)$.*

Доказательство. Поскольку справедливы (1.63), то найдётся такое требование $i \in N'$, что $r_i = r_{\max}(N')$ и $d_i = d_{\min}(N')$. Тогда из (1.57) следует, что $i = j^{n+1}(N')$. Согласно лемме 1.9 найдутся такие множества $N^1, N^2 \subset N'$, что $N^1 \cup N^2 = N' \setminus \{i\}$ и расписание $\pi = (\vec{\pi}^1_r, i, \vec{\pi}^2_d)$, где $\vec{\pi}^1_r \in \vec{\Pi}_r(N^1, t')$, $\vec{\pi}^2_d \in \vec{\Pi}_d(N^2, C_{\max}(\vec{\pi}^1_r, i))$, будет оптимальным. Если $N^1 = \emptyset$, то утверждение леммы справедливо.

Пусть $N^1 \neq \emptyset$. Пусть требование i обслуживается при расписании π k-м по порядку. Тогда $\vec{\pi}^1_r = (j_1, ..., j_{k-1})$, $\vec{\pi}^1_d = (j_{k+1}, ..., j_{n'})$, где $n' = |N'|$, и

$$\pi = (j_1, ..., j_{k-1}, j_k, j_{k+1}, ..., j_{n'}),$$

где $j_k = i$. В силу (1.63) $i \in J_{d_{\min}}(N')$. Тогда согласно лемме 1.5 существует такое $j^* \in J^*(N')$, что $j^* \in N^2 \cup \{j\}$ и, следовательно, справедливо (1.66). Покажем, что

$$L_{\max}(\pi) = L_{j_k}(\pi). \tag{1.72}$$

Поскольку выполняются (1.63), то $r_{j_k} \geq r_j$ для всех $j \in N^2$ и, следовательно, требования j_k, j_{k+1},..., $j_{n'}$ при расписании π обслуживаются без простоев прибора. Это означает, что выполняются равенства (1.68). Согласно (1.63) $d_{j_k} \leq d_j \ \forall \ j \in N^2$. Кроме того, $\vec{\pi}_d^2 \in \vec{\Pi}_d(N^2, C_{\max}(\vec{\pi}_r^1, j_k))$. Следовательно, расписание $(j_k, \vec{\pi}_d^2)$ составлено в порядке неубывания директивных сроков и $d_{j_l} \leq d_{j_{l+1}} \ \forall \ l = \overline{k, n'-1}$. Тогда в силу ограничений (1.63), (1.65) $d_{j_{l+1}} - p_{j_{l+1}} \geq d_{j_l} \ \forall \ l = \overline{k, n'-1}$. Поэтому с учётом (1.68) $L_{j_l}(\pi) = C_{j_l}(\pi) - d_{j_l} \geq C_{j_l}(\pi) + p_{j_{l+1}} - d_{j_{l+1}} = C_{j_{l+1}}(\pi) - d_{j_{l+1}} = L_{j_{l+1}}(\pi) \ \forall \ l = \overline{k, n'-1}$. Отсюда и из (1.66) следует (1.72).

Согласно (1.63) $d_{j_k} \leq d_j \ \forall \ j \in N'$. Следовательно, существует расписание $\pi' \in \vec{\Pi}_d(N', t')$, при котором требование j_k обслуживается первым по порядку. Пусть $\pi' = (i_1, i_2, ..., i_{n'})$ и $i_1 = j_k$. Покажем, что

$$L_{\max}(\pi') = L_{i_1}(\pi'). \tag{1.73}$$

Поскольку выполняются (1.63), то $r_{i_1} \geq r_j \ \forall \ j \in N'$, и требования i_1, i_2,..., $i_{n'}$ при расписании π' обслуживаются без простоев прибора. Следовательно, выполняются равенства

$$C_{i_{l+1}}(\pi') = C_{i_l}(\pi') + p_{i_{l+1}}, \quad l = \overline{1, n'-1}. \tag{1.74}$$

Так как $\pi' \in \vec{\Pi}_d(N', t')$, то

$$d_{i_l} \leq d_{i_{l+1}} \quad l = \overline{1, n'-1}. \tag{1.75}$$

Тогда в силу ограничений (1.63), (1.65) $d_{i_{l+1}} - p_{i_{l+1}} \geq d_{i_l} \ \forall \ l = \overline{k, n'-1}$. Отсюда с учётом (1.74) $L_{i_l}(\pi') = C_{i_l}(\pi') - d_{i_l} \geq C_{i_l}(\pi') + p_{i_{l+1}} - d_{i_{l+1}} = C_{i_{l+1}}(\pi') - d_{i_{l+1}} = L_{i_{l+1}}(\pi') \ \forall \ l = \overline{k, n'-1}$, и равенство (1.73) справедливо.

Очевидно, что $C_{j_k}(\pi) \geq C_{j_k}(\pi')$. Отсюда $L_{j_k}(\pi) \geq L_{j_k}(\pi')$ и, учитывая (1.66), (1.72), (1.73), будем иметь $L_{\max}(\pi) \geq L_{\max}(\pi')$. Таким образом, в результате преобразования оптимального расписания π получили оптимальное расписание π', удовлетворяющее утверждению леммы. Лемма доказана. □

1.8 Оценки абсолютной погрешности

В данном и последующих разделах этой главы предлагается подход получения оценки абсолютной погрешности и нахождения приближённого решения для задач теории расписаний для одного и нескольких приборов с критерием минимизации максимального временно́го смещения.

1.8.1 Задачи для нескольких приборов

На m приборах $M = \{1, \ldots, m\}$ необходимо обслужить требования $j \in N$, где $n = \{1, \ldots, n\}$. Прерывания при обслуживании требований запрещены. В каждый момент времени любой из приборов может обрабатывать не более одного требования. Для каждого требования $j \in N$ заданы: момент возможного начала обслуживания r_j, продолжительность обслуживания $0 \leq p_{ji} \leq +\infty$ (на i-ом приборе)[4] и директивный срок окончания d_j. Между требованиями заданы отношения предшествования в виде ациклического ориентированного графа $G \subset N \times N$.

Через π_i будем обозначать перестановку (расписание) из элементов множества N, обслуживаемых на приборе M_i, $i = 1, \ldots, m$, множество требований в расписании π_i обозначим $\{\pi_i\} \subseteq N$, а через $\pi = \bigcup_{i=1}^{m} \pi_i$, $\{\pi_\alpha\} \bigcap \{\pi_\beta\} = \emptyset$, если $\alpha \neq \beta$, расписание для всего множества требований $N = \{\pi\}$. Естественно, будут рассматриваться только **допустимые** расписания без искусственных простоев приборов, удовлетворяющие отношениям предшествования. Для любого ациклического графа G множество допустимых расписаний не пусто. Момент окончания обслуживания требования j на приборе M_i при расписании π определяется как

$$C_j(\pi) = \max \left\{ r_j, \max_{k \in N_j} C_k(\pi), \max_{(k \to j)_{\pi_i}} C_k(\pi_i) \right\} + p_j, \tag{1.76}$$

где N_j – множество требований предшествующих требованию j, согласно графу G, и $(k \to j)_{\pi_i}$ требования, согласно расписанию π_i на приборе M_i, предшествующих обслуживанию требования $j \in \{\pi_i\}$.

Также под расписанием будем понимать совокупность $S = \{s_j \mid j \in N_i, i \in M\}$ моментов начал обслуживания требований, где $\{N_i \mid i \in M\}$ образуют разбиение[5] множества N. Момент завершения обслуживания требования $j \in N$ в расписании S определим как $C_j(S) = s_j(S) + p_{ji}, j \in N_i$. Расписание S называется *допустимым*, если $r_j \leq s_j(S)$, $C_j(S) < \infty$, $\forall j \in N$; $C_j(S) \leq s_k(S), \forall (j,k) \in G$.

Целевой функцией является минимизация максимального временно́го смещения:

$$\min_{\pi} \max_{j \in N} \{C_j(\pi) - d_j\}.$$

Данная задача $P|prec; r_j|L_{\max}$ является обобщением ряда $NP-$трудных задач, в частности: $P|intree, r_j, p_j = 1|C_{\max}$, $P|outtree, p_j = 1|L_{\max}$ [95];

[4] Если $p_{ji} = +\infty$, то требование j не может быть обслужено на приборе i.

[5] При этом будем полагать, что $\bigcup_{i \in M} N_i = N$, $N_i \bigcap N_l = \emptyset$ при $i \neq l$.

$P2|chains|C_{\max}$ [115]; $P||C_{\max}$ [124]; $1|r_j|L_{\max}$, $P2||C_{\max}$ [168]; $P|prec, p_j = 1|C_{\max}$ [215], для которых выделены полиномиально разрешимые частные случаи.

В данном разделе приведём основные обозначения и определения, которые будут нами использоваться в дальнейшем.

Определение 1.6 *Под примером A исследуемой задачи будем понимать $\{G^A, (r_j^A, p_j^A, d_j^A) | \, j \in N\}$ – некоторый набор параметров требований и графа предшествования. Расписание с минимальным значением максимальнóго временнóго смещения (оптимальное расписание) данного примера будем обозначать через π^A, т.е. $L_{\max}^A(\pi^A) = \min\limits_{\pi} \max\limits_{j \in N} L_j^A(\pi)$, где минимум ищется по множеству всех допустимых расписаний π.*

Определение 1.7 *Под расписанием будем понимать как совокупность перестановок $\pi = \bigcup\limits_{i=1}^{m} \pi_i$, так и вектор моментов начала обслуживания требований $S = \{s_j \, | \, j \in N_i, \ i \in M\}$. Для обозначения оптимальнóго вектора перестановок и оптимальнóго расписания примера A будем использовать π^A и S^A. В раннем расписании каждое требование $j \in N$ начинает обслуживаться в наиболее ранний допустимый момент времени: либо в момент его поступления (r_j^A), либо – сразу после окончания обслуживания предыдущего требования на том же приборе, либо – сразу после окончания обслуживания требования, предшествующему данному (согласно графу G).*

Определение 1.8 *Пусть заданы пример A и некоторое расписание π. Расширенным графом предшествования G_π^A будем называть граф, который получен из графа G^A путём добавления к нему рёбер, согласно перестановкам $\pi_i, \ \ i = 1, \ldots, m$.*

Можно написать критерий допустимости расписания π: расписание π допустимо для примера A, тогда и только тогда, когда расширенный граф предшествования G_π^A без контуров.

Определение 1.9 *Любую цепочку, являющуюся подграфом графа G_π^A будем называть цепочкой предшествования ($\sigma \subset G_\pi^A$).*

Если $\sigma = (j_1, \ldots, j_k) \subset G_\pi^A$ — цепочка предшествования, то в силу формулы (1.76)

$$C_{j_k}^A(\pi) \geq r_{j_1}^A + \sum_{j \in \{\sigma\}} p_j^A. \tag{1.77}$$

Определение 1.10 *Цепочку предшествования* $\sigma = \{j_1, \ldots, j_k\}$ *будем называть задерживающей для требования* j_k *и обозначать* $\sigma^*(j_k)$*, если:*
– требование j_1*, начинает выполняться в момент времени* r_{j_1}*;*
– для некоторого $l = 1, 2, \ldots, k-1$ *требование* j_{l+1} *начинает выполняться в момент завершения обслуживания требования* j_l*.*

Фактически, задерживающая цепочка — это цепочка предшествования для которой неравенство (1.77) превращается в равенство:

$$C^A_{j_k}(\pi) = r^A_{j_1} + \sum_{j \in \{\sigma^*(j_k)\}} p^A_j. \tag{1.78}$$

Задерживающая цепочка существует для каждого требования, т.к. мы рассматриваем только расписания без искусственных простоев.

Аналогично как и для случая одного прибора, определим инверсные примеры.

Определение 1.11 Пример $A = \{G^A, (r^A_j, p^A_j, d^A_j) | \, j \in N\}$ *называется* инверсным *к примеру* $B = \{G^B, (r^B_j, p^B_j, d^B_j) | \, j \in N\}$*, если выполнены соотношения:*

$$\forall \; j \in N \quad r^A_j = -d^B_j, \; p^A_j = p^B_j, \; d^A_j = -r^B_j,$$

а также $(k \to j) \in G^A$ *тогда и только тогда, когда ребро* $(j \to k) \in G^B$*, т.е. в примере* B *ориентация рёбер заменена на противоположную,* $\overleftarrow{G}^B = \overrightarrow{G}^A$*.* Перестановка $\pi'_i = (j_{n_i}, j_{n_i - 1}, \ldots, j_1)$ *называется* инверсной *к перестановке* $\pi_i = (j_1, \ldots, j_{n_i})$ *при обслуживании на приборе* M_i*. Вектор перестановок* π' *называется* инверсным *к перестановке* π*, если все его расписания по соответствующим приборам являются инверсными к расписаниям из* π*.* Отношение предшествования*, инверсное к* G *будем обозначать через* $\overleftarrow{G}$*.* Расписание $S' = \{s'_j \, | \, j \in N_i, \; i \in M\}$ *называется* инверсным *к расписанию* $S = \{s_j \, | \, j \in N_i, \; i \in M\}$*, если* $s'_j = -s_j - p_j, \; \forall \; j \in N_i, \; \forall i \in M$*, где* N_i *– множество требований, обслуживаемых на приборе* M_i*, а* s_j*– момент начала обслуживания требования* j*.*

Определение 1.12 *Расписание* S*, допустимое для заданного примера* $V = \{G^V, (r^V_j, p^V_j, d^V_j) \mid \; j \in N\}$*, будем называть* вполне допустимым*, если каждое требование* $j \in N$ *выполняется в своём директивном интервале* $[r^V_j, d^V_j]$*.*

Определение 1.13 *Для двух произвольных примеров A и B задач $\{P, Q, R\} \mid prec, r_j \mid L_{\max}$ определим следующие функции:*

$$\begin{cases} \rho_d(A,B) = \max\limits_{j\in N}\{d_j^A - d_j^B\} - \min\limits_{j\in N}\{d_j^A - d_j^B\}; \\ \rho_r(A,B) = \max\limits_{j\in N}\{r_j^A - r_j^B\} - \min\limits_{j\in N}\{r_j^A - r_j^B\}; \\ \rho_p(A,B) = \sum\limits_{j\in N}\left(\max\limits_{i\in M}(p_{ji}^A - p_{ji}^B)_+ + \max\limits_{i\in M}(p_{ji}^A - p_{ji}^B)_-\right); \\ \rho(A,B) = \rho_d(A,B) + \rho_r(A,B) + \rho_p(A,B), \end{cases} \tag{1.79}$$

где $(x)_+ = \begin{cases} x, x>0, \\ 0, x\le 0; \end{cases}$; $(x)_- = \begin{cases} -x, x<0, \\ 0, x\ge 0; \end{cases}$; $|x| = (x)_+ + (x)_-$.

Замечание 1.1 *Величину $\rho_p(A,B)$ можно записать по-другому:*

$$\rho_p(A,B) = \sum_{j\in N}\left(\max_{i\in M}\{(p_{ji}^A - p_{ji}^B), 0\} - \min_{i\in M}\{(p_{ji}^A - p_{ji}^B), 0\}\right).$$

Замечание 1.2 *Когда $p_{ji} \in \{p_j, +\infty\}, \forall j \in N, \forall i \in M$, т.е. не зависит от номера прибора, на котором может быть обслужено, тогда*

$$\rho_p(A,B) = \sum_{j\in N} |p_j^A - p_j^B|. \tag{1.80}$$

Необходимо отметить одно важное свойство функции $\rho(A,B)$, – она сепарабельна относительно параметров d, r и p, – это свойство мы в дальнейшем будем использовать.

Определение 1.14 *Пусть задан пример A на множестве требований N (с отношением предшествования G). Будем говорить, как и раньше, что пример B на том же множестве требований наследует у примера A параметр x, если $x_j^B = x_j^A$, $\forall\ j \in N$.*

Напомним без доказательства утверждения из раздела 1.3 с добавлением того, что будем изменять продолжительности обслуживания требований и инвертировать граф отношений предшествования G.

1.8.2 Оценка абсолютной погрешности

Лемма 1.15 *Пусть пример B наследует у примера A все параметры, кроме $\{d_j \mid j \in N\}$. Тогда для любого допустимого (для обоих примеров) расписания π верно соотношение*

$$L_{\max}^B(\pi) - L_{\max}^A(\pi) \le \max_{j\in N}\{d_j^A - d_j^B\}. \tag{1.81}$$

□

Лемма 1.16 *Пусть пример B наследует у примера A все параметры, кроме $\{d_j \mid j \in N\}$, а $\tilde{\pi}^B$ — приближённое решение задачи B, удовлетворяющее условию*[6]

$$0 \le L^B_{\max}(\tilde{\pi}^B) - L^B_{\max}(\pi^B) \le \delta_B, \tag{1.82}$$

где π^B — оптимальное решение, поэтому оно удовлетворяет условию

$$L^B_{\max}(\pi^B) \le L^B_{\max}(\pi), \qquad \forall \pi. \tag{1.83}$$

Тогда

$$0 \le L^A_{\max}(\tilde{\pi}^B) - L^A_{\max}(\pi^A) \le \rho_d(A,B) + \delta_B.$$

□

Замечание 1.3 *Существуют примеры, на которых достигается равенство верхней границы $0 \le L^B_{\max}(\pi^A) - L^B_{\max}(\pi^B) = \rho_d(A,B)$.*

Доказательство. Параметры примеров для $n = 3$, $m = 1, G = \emptyset$:
A: $r^A_j = 0$, $p^A_j = 1, j = 1,2,3$; $d^A_1 = 1, d^A_2 = 2, d^A_3 = 3$.
B: $r^B_j = r^A_j, p^B_j = p^A_j, d^B_j = 2, j = 1,2,3$.

Нетрудно проверить, что
$\pi^A = (1,2,3)$; $L^A_{\max}(\pi^A) = 0$;
$\pi^B = (3,2,1)$; $L^A_{\max}(\pi^B) = 2$;
$\max\limits_{j \in N}\{d^A_j - d^B_j\} = \max\limits_{j \in N}\{d^B_j - d^A_j\} = 1$;
$\rho_d(A,B) = 2$. □

Лемма 1.17 *Пусть V и W – взаимно инверсные примеры со множеством требований N, а π и π' – взаимно инверсные перестановки. Тогда $L^V_{\max}(S^V_\pi) = L^W_{\max}(S^W_{\pi'})$.* □

Следствие 1.5 *Если решение π является оптимальным для примера V, то инверсное решение π' будет оптимальным для примера, инверсного к V.* □

Лемма 1.18 *Пусть пример C наследует у примера B все параметры, кроме $\{r_j \mid j \in N\}$, а $\tilde{\pi}^C$ — приближённое решение задачи C, удовлетворяющее условию*

$$0 \le L^C_{\max}(\tilde{\pi}^C) - L^C_{\max}(\pi^C) \le \delta_C. \tag{1.84}$$

Тогда

$$0 \le L^B_{\max}(\tilde{\pi}^C) - L^B_{\max}(\pi^B) \le \rho_r(B,C) + \delta_C.$$

□

[6] Если $\delta_B = 0$, то $\tilde{\pi}^B$ — оптимальное решение.

Лемма 1.19 *Пусть пример D наследует у примера C все параметры, кроме $\{p_{ji} < \infty \mid j \in N, i \in M\}$. Тогда для любого допустимого расписания π выполняется*

$$L^D_{\max}(\pi) - L^C_{\max}(\pi) \le \sum_{j\in N} \max_{i\in M}(p^D_{ji} - p^C_{ji})_+.$$

Доказательство. Имеем $L^D_{\max}(\pi) = \max\limits_{j\in N}\{C^D_j(\pi) - d^D_j\}$. Пусть j^* – требование, на котором этот максимум достигается (на приборе i^*), т.е. $L^D_{\max}(\pi) = C^D_{j^*}(\pi) - d^D_{j^*}$. Очевидно

$$L^C_{\max}(\pi) \ge C^C_{j^*}(\pi) - d^C_{j^*} = s_{j^*}(\pi) + p^C_{j^*i^*} - d^C_{j^*}.$$

Очередное требование не может начаться раньше, чем закончится предыдущее, следовательно $s_j \ge C^C_k(\pi)$, если требование k предшествует j (по отношению предшествования G или в перестановке на том же приборе i^*). Значит, $L^C_{\max}(\pi) \ge s_{j_1} + \sum\limits_{j_k\in\{\sigma^*(j^*)\}} p^C_{j_ki_k} - d^C_{j^*}$, где j_1 – первое требование в задерживающей цепочке $\sigma^*(j^*) = (j_1, \dots, j^*)$. Далее,

$$L^C_{\max}(\pi) \ge s_{j_1} + \sum_{j_k\in\{\sigma^*(j^*)\}} p^D_{j_ki_k} - d^D_{j^*} - \sum_{j_k\in\{\sigma^*(j^*)\}} (p^D_{j_ki_k} - p^C_{j_ki_k}) \ge$$

$$\ge L^D_{\max}(\pi) - \sum_{j_k\in\{\sigma^*(j^*)\}} (p^D_{j_ki_k} - p^C_{j_ki_k})_+ \ge L^D_{\max}(\pi) - \sum_{j\in N}(p^D_{ji} - p^C_{ji})_+.$$

Отсюда следует утверждение леммы 1.19. □

Лемма 1.20 *Пусть пример D наследует у примера C все параметры, кроме $\{p_{ji} \mid j \in N, i \in M\}$, а $\tilde\pi^D$ – приближённое решение задачи D, удовлетворяющее условию*

$$L^D_{\max}(\tilde\pi^D) - L^D_{\max}(\pi^D) \le \delta_D. \tag{1.85}$$

Тогда

$$0 \le L^C_{\max}(\tilde\pi^D) - L^C_{\max}(\pi^C) \le \rho_p(C, D) + \delta_D.$$

Доказательство. Схема доказательства аналогична схеме доказательства леммы 1.16, поэтому приведём только выкладки:

$$\begin{aligned} &L^C_{\max}(\tilde\pi^D) - L^C_{\max}(\pi^C) \le \\ \le &L^D_{\max}(\tilde\pi^D) - L^C_{\max}(\pi^C) + \sum_{j\in N} \max_{i\in M}(p^C_{ji} - p^D_{ji})_+ \le \end{aligned}$$

$$\leq \delta_D + L^D_{\max}(\pi^D) + \sum_{j\in N} \max_{i\in M}(p^C_{ji} - p^D_{ji})_+ - L^D_{\max}(\pi^C) +$$
$$+ \sum_{j\in N} \max_{i\in M}(p^D_{ji} - p^C_{ji})_+ \leq$$
$$\leq \delta_D + \sum_{j\in N} \left(\max_{i\in M}(p^C_{ji} - p^D_{ji})_+ + \max_{i\in M}(p^C_{ji} - p^D_{ji})_- \right) = \delta_D + \rho_p(C, D).$$

□

Замечание 1.4 *Оценка, полученная в лемме 1.20, неулучшаема.*

Доказательство. Параметры примеров для $n = 1$, $m = 2$:
A: $r^A_1 = d^A_1 = 0$; $p^A_{11} = 1, p^A_{12} = 3$.
B: $r^B_1 = d^B_1 = 0, p^B_{11} = p^B_{12} = 2$.
Нетрудно проверить, что:
$\pi^A = \{(1), (\emptyset)\}$[7]; $L^A_{\max}(\pi^A) = 1$;
$\pi^B = \{(\emptyset), (1)\}$; $L^A_{\max}(\pi^B) = 3$;
$\max\limits_{i\in M}(p^A_{1i} - p^B_{1i})_+ = \max\limits_{i\in M}(p^A_{1i} - p^B_{1i})_- = 1$;
$\rho_p(A, B) = 2$. □

Теорема 1.7 *Пусть пример D наследует у примера A все параметры, кроме $\{d_j, r_j, p_{ji} \mid j \in N, i \in M\}$, а $\tilde{\pi}^D$ – приближённое решение примера D, удовлетворяющее условию (1.85). Тогда*

$$0 \leq L^A_{\max}(\tilde{\pi}^D) - L^A_{\max}(\pi^A) \leq \rho(A, D) + \delta_D. \tag{1.86}$$

Доказательство. Неравенство $0 \leq L^A_{\max}(\tilde{\pi}^D) - L^A_{\max}(\pi^A)$ вытекает из оптимальности решения π^A для примера A. Докажем неравенство

$$L^A_{\max}(\tilde{\pi}^D) - L^A_{\max}(\pi^A) \leq \rho(A, D) + \delta_D.$$

Воспользуемся аддитивным свойством оценок, полученных в леммах 1.16, 1.18 и 1.20. Рассмотрим вспомогательные примеры B и C с параметрами требований:

$$\begin{array}{c} d^A_j, \quad d^B_j = d^C_j = d^D_j; \\ r^A_j = r^B_j, \quad r^C_j = r^D_j; \\ p^A_{ji} = p^B_{ji} = p^C_{ji}, \quad p^D_{ji}. \end{array}$$

Тогда:
$\rho(A, B) = \rho_d(A, B)$; $\rho(B, C) = \rho_r(B, C)$; $\rho(C, D) = \rho_p(C, D)$.

[7] $\pi = \{\pi_1, \pi_2\}$ означает, что π_1 – расписание для первого прибора, а π_2 – для второго.

Применяя последовательно леммы 1.20, 1.18 и 1.16, и полагая $\tilde{\pi}^D = \tilde{\pi}^C$, $\delta_C = \delta_D + \rho_p(C, D)$, $\tilde{\pi}^C = \tilde{\pi}^B$, $\delta_B = \delta_C + \rho_r(B, C)$, получим $0 \le L^A_{\max}(\tilde{\pi}^D) - L^A_{\max}(\pi^A) \le \rho_d(A, B) + \rho_r(B, C) + \rho_p(C, D) + \delta_D = \rho_d(A, D) + \rho_r(A, D) + \rho_p(A, D) + \delta_D = \rho(A, D) + \delta_D$.
Теорема 1.7 доказана. □

Замечание 1.5 *Оценка, полученная в теореме 1.7, неулучшаема.*

Доказательство. Параметры примеров для $n = 4$, $m = 1$:
A: $r^A_1 = r^A_3 = r^A_4 = 0$, $r^A_2 = 1$; $p^A_j = 1, j = 1, 2, 3, 4$; $d^A_j = j, j = 1, 2, 3, 4$;
B: $r^B_1 = 1$, $r^B_j = r^A_j, j = 2, 3, 4$; $p^B_j = p^A_j, j = 1, 2, 3, p^B_4 = 0$; $d^B_1 = 2$,
$d^B_j = d^A_j, j = 2, 3, 4$.

Нетрудно проверить, что:
$\pi^A = (1, 2, 3, 4)$; $L^A_{\max}(\pi^A) = 0$;
$\pi^B = (4, 3, 2, 1)$; $L^A_{\max}(\pi^B) = 3$;
$\rho_d(A, B) = \rho_r(A, B) = \rho_p(A, B) = 1$;
$\rho(A, B) = 3$. □

Для заинтересованного читателя хотелось бы предложить задачу: построить примеры C и B, на которых, оценка, полученная в лемме (1.86), неулучшаема.

1.8.3 Схема приближённого решения задачи $P \mid prec, r_j \mid L_{\max}$

Нахождение приближённого решения рассматриваемой NP-трудной задачи состоит из двух шагов. На первом шаге для исходного примера $A = \{G, (r^A_j, p^A_j, d^A_j) | \, j \in N\}$ мы изменяем параметры $\{(r^A_j, p^A_j, d^A_j) | \, j \in N\}$ таким образом, чтобы полученный пример $B = \{G, (r^B_j, p^B_j, d^B_j) | \, j \in N\}$ (граф предшествования G одинаков для обоих примеров) принадлежал некоторому полиномиально разрешимому случаю исходной задачи. На следующем шаге находим оптимальное расписание для примера B. Согласно теореме 1.7, оптимальное расписание π^B для примера A будет иметь оценку абсолютной погрешности $0 \le L^A_{\max}(\pi^B) - L^A_{\max}(\pi^A) \le \rho(A, B)$.

Рассмотрим полиномиально разрешимый случай исходной задачи, когда параметры требований удовлетворяют k линейно независимым неравенствам

$$XR + YP + ZD \le H, \tag{1.87}$$

(при естественном ограничении $p_j \ge 0$, $\forall j \in N$), где $R = (r_1, \ldots, r_n)^T$, $P = (p_1, \ldots, p_n)^T$, $D = (d_1, \ldots, d_n)^T$ и X, Y, Z – матрицы размерности $k \times n$, а

$H = (h_1, \ldots, h_k)^T$ – k-мерный вектор (верхний индекс ^{T} обозначает операцию транспонирования). Затем в этом классе примеров (1.87) находим пример B с минимальным "расстоянием" $\rho(A, B)$ до исходного примера A, решая следующую задачу

$$\begin{cases} (x^d - y^d + x^r - y^r) + \sum\limits_{j \in N} x_j^p \longrightarrow \min \\ y^d \le d_j^A - d_j^B \le x^d, \quad \forall j \in N, \\ y^r \le r_j^A - r_j^B \le x^r, \quad \forall j \in N, \\ -x_j^p \le p_j^A - p_j^B \le x_j^p, \quad \forall j \in N, \\ 0 \le x_j^p, \quad \forall j \in N, \\ XR^B + YP^B + ZD^B \le H. \end{cases} \tag{1.88}$$

Необходимо заметить, все известные автору полиноминально разрешимые случаи NP– трудных задач теории расписаний могут быть представлены системой линейных неравенств вида (1.87). Мы фактически проецируем исходный пример (точку) A на разрешимый класс примеров (1.87).

Задача линейного программирования (1.88) с $3n + 4 + n$ переменными ($r_j^B, p_j^B, d_j^B, j = 1, \ldots, n$, и x_d, y_d, x_r, y_r, и x_j^p, $j = 1, \ldots, n$) и $7n + k$ неравенствами иногда может быть решена за полиномиальное (от n и k) операций, учитывая специфику ограничений задачи (1.88). Например, для $NP-$трудной в сильном смысле задачи $1|r_j|L_{\max}$ имеются два нетривиальных полиномиально разрешимых случая:

$$\begin{cases} d_1 \le \ldots \le d_n, \\ d_1 - r_1 - p_1 \ge \ldots \ge d_n - r_n - p_n, \end{cases} \tag{1.89}$$

и

$$\max_{k \in N} \{d_k - r_k - p_k\} \le d_j - r_j, \, \forall j \in N. \tag{1.90}$$

В случае (1.89) оптимальное решение задачи $1|r_j|L_{\max}$ может быть найдено за $O(n^3 \log n)$ операций (раздел 2.1). А задача линейного программирования (1.88) может быть решена за $O(n \log n)$ операций (раздел 1.4.1).

В случае же (1.90) оптимальное расписание находится за $O(n^2 \log n)$ операций [130]. Как и в случае (1.89) минимум абсолютной погрешности максимального временно́го смещения находится за полиномиальное время, в данном случае за $O(n)$ операций (раздел 1.4.2).

В главе 3 будут приведены ещё два новых полиномиально разрешимых подслучая исследуемой NP-трудной задачи:

$$r_i \le r_j \Rightarrow d_i \ge d_j \; \forall \, i, j \in N;$$

$$d_j - p_j \le d_{\min}(N)\ \forall\, j \in N,$$

и

$$r_i \le r_j \Rightarrow d_i - p_i \ge d_j\ \forall\, i, j \in N, \quad i \ne j.$$

В первом случае будет представлен алгоритм решения трудоёмкости $O(n^2)$ опрераций, во втором – $O(n \log n)$ операций.

В случае, когда для исходной задачи нет полиномиально разрешимых выделенных подслучаев задачи (в скобках заметим, что тривиальные подслучаи задачи обычно не позволяют найти качественно новые оценки абсолютной погрешности) либо "расстояние" $\rho(A, C)$ до любого полиномиально разрешимого примера C "слишком велико". Но для некоторого примера $B = \{G, (r_j^B, p_j^B, d_j^B) |\, j \in N\}$ оценка абсолютной погрешности максимального временно́го смещения приближённого расписания $\tilde{\pi}$ является "приемлемой", тогда для исходного примера $A = \{G, (r_j^A, p_j^A, d_j^A) |\, j \in N\}$ приближённое расписание $\tilde{\pi}$ имеет гарантированную абсолютную погрешность от оптимального значения целевой функции, согласно теореме 1.7, не превышающую $\delta^B(\tilde{\pi}) + \rho(A, B)$. Величина $\delta^B(\tilde{\pi}) + \rho(A, B)$ порой оказывается существенно меньше, чем $\rho(A, C)$ для любого полиномиально/псевдополиномиально разрешимого примера C.

1.9 Нормированное пространство примеров

Рассмотрим совокупность примеров из класса задач $P|prec, r_i|L_{\max}$ с фиксированным количеством требований n, количеством приборов m и графом предшествования G. Данная совокупность примеров образует $3n$-мерное линейное пространство ($3n$ параметров: $r_j, p_j, d_j, j = \overline{1, n}$).

Определение 1.15 *Два примера* $A = \{G, (r_j^A, p_j^A, d_j^A) |\, j \in N\}$ *и* $B = \{G, (r_j^B, p_j^B, d_j^B) |\, j \in N\}$ *назовём эквивалентными, если*

$$\exists\, d, r: \quad d_j^A = d_j^B + d, \quad r_j^A = r_j^B + r, \quad p_j^A = p_j^B \qquad \forall j \in N.$$

Данное определение порождает разбиение рассматриваемой совокупности примеров задачи на классы эквивалентности. Для удобства в качестве представителя класса выберем такой пример, у которого $r_1 = 0$ и $d_1 = 0$. Полученное семейство классов эквивалентности является $(3n-2)$-мерным линейным пространством. Обозначим его через $\mathcal{L}_n$. Будем говорить, что пример A принадлежит классу $\mathcal{L}_n$, если выполняется условие

$$r_1 = d_1 = 0. \tag{1.91}$$

Замечание 1.6 *У эквивалентных примеров совпадают множества оптимальных расписаний (перестановок).* □

Теперь на пространстве классов эквивалентности примеров рассмотрим следующий функционал

$$\forall A \in \mathcal{L}_n \quad \varphi(A) = \max_{j\in N}(r_j^A) - \min_{j\in N}(r_j^A) + \max_{j\in N}(d_j^A) - \min_{j\in N}(d_j^A) + \sum_{j\in N}|p_j^A| \geq 0.$$

Данный функционал удовлетворяет трём свойствам нормы:

$$\begin{cases} \varphi(A) = 0 \Longleftrightarrow A \equiv 0; \\ \varphi(\alpha A) = \alpha\varphi(A); \\ \varphi(A+B) \leq \varphi(A) + \varphi(B). \end{cases} \tag{1.92}$$

$A = \emptyset$, когда $r_j = p_j = d_j = 0, \forall j \in N$. Первое свойство следует из определения функционала $\varphi(A)$, второе проверяется непосредственно, а третье следует из того, что максимум суммы не превосходит суммы максимумов и модуль суммы не превосходит суммы модулей, – мы опять использовали сепарабельность функционала.

Таким образом, $\mathcal{L}_n$ является $(3n-2)$-мерным линейным нормированным пространством с нормой $||A|| = \varphi(A)$. Необходимо заметить, что данная норма естественным образом порождает метрику, определённую в (1.79), (1.80): $\rho(A,B) = ||A - B||$.

Рассмотрим некоторые свойства функционала $\varphi(A) = ||A||$. Чтобы задать топологию пространства, достаточно задать систему окрестностей нуля, в данном случае – это "шары" с центром в точке ноль. Опишем теперь единичный шар в пространстве примеров задачи с изменяющимися параметрами r_i и d_i. Единичный шар – это множество точек (примеров) A, удовлетворяющие условию

$$||A|| \leq 1. \tag{1.93}$$

Рассмотрим случай $n = 2$. Условие (1.93) перепишется в виде

$$|r_2| + |d_2| \leq 1 \quad \text{(см. рис. 1.6)}.$$

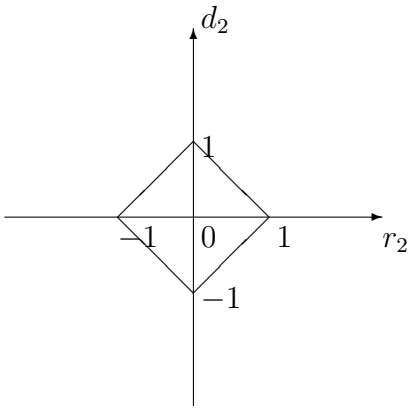

Рис. 1.6: $n = 2$

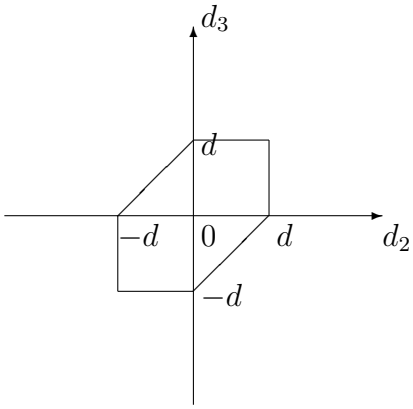

Рис. 1.7: $n = 3$

Теперь рассмотрим случай $n = 3$. Условие (1.93) можно переписать в виде:

$$\begin{cases} r + d \le 1; \\ r \ge 0; \\ d \ge 0; \\ -d \le d_2 \le d; \\ -d \le d_3 \le d; \\ -d \le d_3 - d_2 \le d; \\ -r \le r_2 \le r; \\ -r \le r_3 \le r; \\ -r \le r_3 - r_2 \le r. \end{cases}$$

Проекция единичного шара на плоскость, параллельную плоскости (d_2, d_3) приведена на рис. 1.7.

Для произвольного n условие (1.93) эквивалентно системе неравенств:

$$\begin{cases} r + d \le 1; \\ r \ge 0; \\ d \ge 0; \\ p_i \ge 0, \quad 1 \le i \le n; \\ -d \le d_i \le d, \quad 2 \le i \le n; \\ -d \le d_i - d_j \le d, \quad 2 \le i < j \le n; \\ -r \le r_i \le r, \quad 2 \le i \le n; \\ -r \le r_i - r_j \le r, \quad 2 \le i < j \le n. \end{cases}$$

1.10 Схемы нахождения приближённого решения

В данном разделе покажем как можно использовать описанный выше подход для нахождения приближённого решения с оценкой абсолютной погрешности задач теории расписаний. Пусть необходимо решить пример A задачи $\alpha \mid \beta \mid L_{\max}$, будем использовать обозначение $\alpha^A \mid \beta^A \mid L_{\max}$. Необходимо так изменить параметры приборов и требований примера A: α^A, β^A и, возможно, целевую функцию $L_{\max}$ на $C_{\max}$, чтобы в результате был построен пример B задачи $\alpha^B \mid \beta^B \mid \{L_{\max}, C_{\max}\}$, для которого мы можем найти приближённое (или точное) решение $\tilde{\pi}^B$ и затем его применить для исходного примера A задачи $\alpha^A \mid \beta^A \mid L_{\max}$.

1.10.1 Схема сведения задач $\alpha \mid \beta \mid L_{\max} \to \alpha \mid \beta \mid C_{\max}$ и $\alpha \mid \beta, r_j \mid L_{\max} \to \alpha \mid \beta, r_j = 0 \mid L_{\max}$

Пусть имеется некоторый пример A задачи $\alpha^A|\beta^A|L_{\max}$, относящейся к классу NP-трудных задач, известно точное или приближённое решение (полученное за полиномиальное/псевдополиномиальное количество операций) $\tilde{\pi}^B$ примера задачи B: $\alpha^A|\beta^A|C_{\max}$ с абсолютной погрешностью, не превосходящей $\delta_B \ge 0$.

В примере задачи B имеем $d_j^B = 0, \forall j \in N$, поэтому из леммы 1.16 получаем оценку:

$$0 \le L_{\max}^A(\tilde{\pi}^B) - L_{\max}^A(\pi^A) \le \rho_d(A, B) + \delta_B = \max_{j \in N} d_j^A - \min_{j \in N} d_j^A + \delta_B.$$

Фактически полученная оценка позволяет оценить "переход" от целевой функции $L_{\max}$ к $C_{\max}$ (*makespane*).

Для приближённого решения задачи $\alpha \mid \beta, r_j \mid L_{\max}$ можно использовать решение задачи $\alpha \mid \beta, r_j = 0 \mid L_{\max}$. В этом случае получаем оценку абсолютной погрешности приближённого решения $\tilde{\pi}^B$, согласно лемме 1.18:

$$0 \le L^A_{\max}(\tilde{\pi}^B) - L^A_{\max}(\pi^A) \le \max_{j \in N} r^A_j - \min_{j \in N} r^A_j + \delta_B.$$

Данная оценка устанавливает связь задач минимизации $L_{\max}$ при одновременном и неодновременном поступлении требований на обслуживание.

1.10.2 Схема сведения задач $\alpha \mid \beta \mid L_{\max} \to \alpha \mid \beta \mid C_{\max}$ и $\alpha \mid \beta, r_j \mid L_{\max} \to \alpha \mid \beta, r_j = 0 \mid L_{\max}$

Пусть необходимо решить пример A задачи $\alpha^A|\beta^A|L_{\max}$, относящейся к классу NP-трудных задач, а также известно точное или приближённое решение $\tilde{\pi}^B$ примера задачи B: $\alpha^A|\beta^A, p_j = p|L_{\max}$ с абсолютной погрешностью, не превосходящей $\delta_B \ge 0$.

Для того, чтобы абсолютная погрешность полученного решения была наименьшей, необходимо найти оптимальное значение параметра p, при котором расстояние $\rho_p(A, B)$ между примерами было минимальным

$$\rho_p(A, B) = \sum_{j=1}^{n} |p^A_j - p| \to \min_p. \tag{1.94}$$

Данная функция является непрерывной, выпуклой, кусочно-линейной функцией относительно p. Если n – нечётное, то минимум достигается на "среднем" p^A_j, т.е. если все p^A_j упорядочены по неубыванию значений, то решение задачи (1.94) $p^* = p^A_{\frac{n+1}{2}}$. Если же n – чётное, то минимум достигается на двух "средних" p^A_j, а также на любом значении между ними, т.е. $p^* \in [p^A_{\frac{n}{2}}; p^A_{\frac{n}{2}+1}]$. Таким образом, $p^* = p^A_{\lfloor \frac{n+1}{2} \rfloor}$ и, согласно лемме 1.20, будем иметь

$$0 \le L^A_{\max}(\tilde{\pi}^B) - L^A_{\max}(\pi^A) \le \sum_{j=1}^{n} \left| p^A_j - p^A_{\lfloor \frac{n+1}{2} \rfloor} \right| + \delta_B.$$

Схема сведения $\alpha \mid \beta, p_j \mid L_{\max} \to \alpha \mid \beta, p_j = 1 \mid L_{\max}$.

Пусть имеется пример A задачи $\alpha^A|\beta^A|L_{\max}$, относящейся к классу NP-трудных задач, известно точное или приближённое решение $\tilde{\pi}^B$ задачи B: $\alpha^A|\beta^A, p_j = p|L_{\max}$ с абсолютной погрешностью, не превосходящей $\delta_B \ge 0$. Условие $p_j = 1$ надо понимать не буквально, это означает, что $p_j = p$, а все параметры r^A_j и d^A_j кратны числу p. В качестве p можно взять $p = p^A_{\lfloor \frac{n+1}{2} \rfloor}$,

а чтобы все r_j^A и d_j^A были кратны числу p, надо вычесть из них остаток от деления этих параметров на p. Тогда $\rho_r(A,B) \le p$ и $\rho_d(A,B) \le p$. Для примера B будем иметь: $p_j^B = p = p^A_{\lfloor \frac{n+1}{2} \rfloor}$, $r_j^B = r_j^A - (r_j^A \ mod \ p)$, $d_j^B = d_j^A - (d_j^A \ mod \ p)$, $\forall j \in N$, и оценка абсолютной погрешности

$$0 \le L^A_{\max}(\tilde{\pi}^B) - L^A_{\max}(\pi^A) \le \sum_{j \in N} \left| p_j^A - p^A_{\lfloor \frac{n+1}{2} \rfloor} \right| + 2p^A_{\lfloor \frac{n+1}{2} \rfloor}.$$

1.10.3 Схема сведения задач $\{R,Q\} \mid \beta \mid L_{\max} \to P \mid \beta \mid L_{\max}$

В задачах $\{R,Q\} \mid \beta \mid L_{\max}$ в отличие от задачи $P \mid \beta \mid L_{\max}$ приборы не идентичны, т.е. продолжительности обслуживания требования $j \in N$ на разных приборах могут различаться: $p_{ji} \ne p_{jk}, i \ne k, i,k \in M$.

Пусть имеется пример A задачи $R|\beta^A|L_{\max}$, относящейся к классу NP-трудных задач, известно точное или приближённое решение $\tilde{\pi}^B$ задачи B $P|\beta^A, p_{ji} \in \{p_j, \infty\}|L_{\max}$ с абсолютной погрешностью, не превосходящей $\delta_B \ge 0$.

Для того, чтобы абсолютная погрешность полученного решения $\tilde{\pi}^B$ была наименьшей, необходимо найти значения параметров p_j^B, $j \in N$, которые бы минимизировали расстояние между примерами A и B

$$\rho_p(A,B) = \sum_{j \in N} \left(\max_{i \in M}\{(p_{ji}^A - p_j^B), 0\} - \min_{i \in M}\{(p_{ji}^A - p_j^B), 0\} \right) \to \min_{p_j^B, \forall j \in N}.$$

Известно

$$\rho_p(A,B) = \sum_{j \in N} \left(\max_{i \in M}\{p_{ji}^A, p_j^B\} - \min_{i \in M}\{p_{ji}^A, p_j^B\} \right) = \sum_{j \in N} \left(\max_{i \in M} p_{ji}^A - \min_{i \in M} p_{ji}^A \right)$$

при условии, что $p_j^B \in [\max\limits_{i \in M} p_{ji}^A, \min\limits_{i \in M} p_{ji}^A]$ $\forall j \in N$. Поэтому

$$0 \le L^A_{\max}(\tilde{\pi}^B) - L^A_{\max}(\pi^A) \le \sum_{j \in N} \left(\max_{i \in M} p_{ji}^A - \min_{i \in M} p_{ji}^A \right).$$

1.10.4 Схема сведения задачи $R \mid \beta \mid L_{\max} \to Q \mid \beta \mid L_{\max}$

Пусть имеется пример A задачи $R|\beta|L_{\max}$ $(p_{ji} < \infty)$, относящейся к классу NP-трудных задач, известно точное или приближённое решение $\tilde{\pi}^B$ задачи B $Q|\beta|L_{\max}$ с абсолютной погрешностью, не превосходящей $\delta_B \ge 0$. В задаче

$Q|\beta|L_{\max}$ длительности обслуживания требований вычисляются по формуле $p_{ji} = \sigma_i p_j, \forall j \in N, \forall i \in M$, где σ_i — производительность i-го прибора.

Теперь для нахождения оценки абсолютной погрешности решения $\tilde{\pi}^B$ необходимо решить задачу:

$$\rho_p(A,B) = \sum_{j\in N}\left(\max_{i\in M}\{(p_{ji}^A - p_j^B\sigma_i^B), 0\} - \min_{i\in M}\{(p_{ji}^A - p_j^B\sigma_i^B), 0\}\right) \to \min_{p_j^B,\sigma_i^B}.$$

Эту задачу можно записать в виде задачи линейного программирования:

$$\begin{cases} \sum\limits_{j\in N}(\alpha_j - \beta_j) \to \min\limits_{\alpha_j,\beta_j,p_j^B,\sigma_i^B} \\ \beta_j \le p_{ji}^A - p_j^B\sigma_i^B \le \alpha_j, \qquad j \in N, i \in M; \\ \beta_j \le 0 \le \alpha_j, \qquad j \in N. \end{cases} \tag{1.95}$$

1.10.5 Схема сведения задачи $\alpha \mid \beta \mid L_{\max} \to \alpha|\beta, p_j \in \{p_1, \ldots, p_k\}|C_{\max}$

Пусть имеется пример A задачи $\alpha^A|\beta^A|L_{\max}$ из класса NP-трудных задач, известно точное или приближённое решение $\tilde{\pi}^B$ задачи B: $\alpha^A|\beta^A, p_j \in \{p_1^B, \ldots, p_k^B\}|C_{\max}$ с абсолютной погрешностью, не превосходящей $\delta_B \ge 0$. В этом случае будем иметь оценку абсолютной погрешности

$$0 \le L_{\max}^A(\tilde{\pi}^B) - L_{\max}^A(\pi^A) \le \rho_p(A,B) + \rho_d(A,B) + \delta_B,$$

где $\rho_d(A,B) = \max\limits_{j\in N} d_j^A - \min\limits_{j\in N} d_j^A$, а для нахождения $\rho_p(A,B)$ необходимо решить следующую задачу:

$$\sum_{j=1}^{n} \min_{1\le l\le k} \left|p_j^A - p_l^B\right| \to \min_{p_1^B,\ldots,p_k^B}.$$

Оставим читателю возможность решения полученной оптимизационной задачи.

Применение схем

Рассмотрим несколько примеров применения вышеуказанных схем. Их можно использовать как отдельно, так и в совокупности.

Один прибор

Задача $1|r_j|L_{\max}$ является NP-трудной в сильном смысле [168]. Схемы нахождения приближённых решений приведены в [49]. Более сложная задача $1|prec, r_j|L_{\max}$ также является NP-трудной в сильном смысле, для неё есть два полиномиально разрешимых случая: $1|prec, r_j|C_{\max}$ [144] и

$1|prec, p_j = p, r_j|L_{\max}$ [202]. Эту задачу можно приближённо решить путём применения одного из двух алгоритмов, в зависимости от того, где меньше абсолютная погрешность.

Два параллельных прибора

Задача $P2|chains|C_{\max}$ является NP-трудной в сильном смысле [115]. Существуют полиномиальные алгоритмы решения следующих "близких" задач, например: $P2|p_j = p, prec|L_{\max}$ [122], $P2|p_j = 1, prec, r_j|L_{\max}$ [123] и $Q2|p_j = p, chains|C_{\max}$ [96].

Произвольное число параллельных приборов

Следующие задачи являются NP-трудными в сильном смысле:
$P||C_{\max}$ [124], $P|p_j = 1, intree, r_j|C_{\max}$, $P|p_j = 1, outtree|L_{\max}$ [95],
$P|p_j = 1, prec|C_{\max}$ [215], $Q|p_j = p, chains|C_{\max}$ [140].

Полиномиально разрешимые случаи: $P|p_j = p, r_j|L_{\max}$ [203],
$P|p_j = p, tree|C_{\max}$ [110, 132], $P|p_j = p, outtree, r_j|C_{\max}$ [95],
$P|p_j = p, intree|L_{\max}$ [95], $P|p_j = 1, chains, r_j|L_{\max}$ [86].

Для задач с отношением предшествования произвольного вида нет точных полиномиально разрешимых случаев. На наш взгляд, важным является вопрос "модификации" графа предшествования и получения оценок погрешности в результате такой "модификации".

Данный подход может быть применён и для построения приближённого решения и получения оценки абсолютной погрешности для задач конвейрного типа (задачи $Flow - shop$).

Следующие задачи являются NP-трудными в сильном смысле:
$F2||L_{\max}$ [168], $F2|r_j|C_{\max}$ [168], $F2|chains|C_{\max}$ [168], $F3||C_{\max}$ [124],
$F|p_{ji} = 1, prec|C_{\max}$ [169, 214], $F|p_{ji} = 1, intree, r_j|C_{\max}$ [97],
$F|p_{ji} = 1, outtree|L_{\max}$ [97].

Полиномиально разрешимые случаи: $F2||C_{\max}$ [135],
$F2|p_{ji} = 1, prec, r_j|L_{\max}$, $F|p_{ji} = 1, tree|C_{\max}$, $F|p_{ji} = 1, intree|L_{\max}$,
$F|p_{ji} = 1, outtree, r_j|C_{\max}$[100].

В конце главы сделаем несколько замечаний.

Таким образом, важным является вопрос нахождения новых эффективных полиномиально разрешимых частных случаев исследуемых NP-трудных задач.

Полученные оценки могут быть использованы при решении исследуемых задач алгоритмами типа метода ветвей и границ (Constraint Programming, branch and cuts, branch and price...).

Важным является вопрос соотношения величин $\rho(A, D)$ и $L^A_{\max}(\pi^D) - L^A_{\max}(\pi^A)$ в формуле (1.87) в случае, когда $\delta_D = 0$.

Насколько теоретическая оценка $\rho(A, D)$ отличается от практической оценки $L^A_{\max}(\pi^D) - L^A_{\max}(\pi^A)$?

Чему равно

$$\rho(A) = \max_D \frac{\rho(A, D)}{L^A_{\max}(\pi^D) - L^A_{\max}(\pi^A)} \, ?$$

Гипотеза: Для любого примера A задачи $1|r_j|L_{\max}$ выполняется $\rho(A) \leq 2$.

Глава 2

Полиномиально и псевдополиномиально разрешимые случаи задачи $1 \mid r_j \mid L_{\max}$

2.1 Задача $1 \mid d_i \leq d_j, d_i - r_i - p_i \geq d_j - r_j - p_j \mid L_{\max}$

Кроме задачи $1 \mid r_j \mid L_{\max}$ в данном параграфе рассматривается задача построения Парето-оптимального множества расписаний по критериям $C_{\max}$ и $L_{\max}$ – задача $1|r_j|L_{\max}$, $C_{\max}$. Будет сформулирован алгоритм построения множества расписаний $\Phi(N,t) = \{\pi'_1, \pi'_2, \ldots, \pi'_m\}$, для которых

$$C_{\max}(\pi'_1) < C_{\max}(\pi'_2) < \ldots < C_{\max}(\pi'_m),$$

$$L_{\max}(\pi'_1) > L_{\max}(\pi'_2) > \ldots > L_{\max}(\pi'_m).$$

И не существует расписания π такого, что $C_{\max}(\pi) \leqslant C_{\max}(\pi'_i)$ и $L_{\max}(\pi) \leqslant L_{\max}(\pi'_i)$, хотя бы одно из неравенств строгое для некоторого i, $i = \overline{1,m}$. Будет показано, что $m \leqslant n$.

2.1.1 Свойства задачи

Как и ранее будем обозначать предшествование требования i требованию j при расписании π как $(i \to j)_\pi$. Введём также:

$$r_j(t) = \max\{r_j, t\}; \tag{2.1}$$

$$r(N,t) = \min_{j \in N}\{r_j(t)\}. \tag{2.2}$$

В тех случаях, когда очевидно о каком множестве требований идёт речь, будем записывать вместо $r(N,t)$ просто $r(t)$.

Пусть π – расписание обслуживания требований некоторого подмножества из N, для определения которого как и раньше используется обозначение $\{\pi\}$.

Предполагается, что параметры требований удовлетворяют ограничениям:

$$d_1 \le \ldots \le d_n, \quad d_1 - r_1 - p_1 \ge \ldots \ge d_n - r_n - p_n. \tag{2.3}$$

Например, данным ограничениям соответствует случай, когда $d_j = r_j + p_j + z$, $j = 1, \ldots, n$, где z – константа, т.е. когда все требования имеют одинаковый запас времени до своего директивного срока.

Допустим $\mid N \mid > 1$ и t – момент освобождения прибора. Выделим из множества N два требования $f = f(N,t)$ и $s = s(N,t)$ следующим образом:

$$f(N,t) = \arg\min_{j \in N} \{d_j \mid r_j(t) = r(N,t)\}, \tag{2.4}$$

$$s(N,t) = \arg\min_{j \in N \setminus \{f\}} \{d_j \mid r_j(t) = r(N \setminus f, t)\}, \tag{2.5}$$

где $f = f(N,t)$. Если $N = \{i\}$, то полагаем $f(N,t) = i$, $s(N,t) = 0$, $\forall t$. Также определим $d_0 = +\infty$, $f(\emptyset, t) = 0$, $s(\emptyset, t) = 0$, $\forall t$. Для требований f и s выполняются следующие свойства.

Лемма 2.1 *Если для требований множества N справедливо (2.3), то при любом расписании $\pi \in \Pi(N)$ для всех $j \in N \setminus \{f\}$, для которых $(j \to f)_\pi$ имеет место*

$$L_j(\pi) < L_f(\pi) \tag{2.6}$$

и для всех $j \in N \setminus \{f, s\}$, отвечающих условию $(j \to s)_\pi$, выполняется

$$L_j(\pi) < L_s(\pi), \tag{2.7}$$

где $f = f(N,t)$ и $s = s(N,t)$

Доказательство. Для всех требований j таких, что $(j \to f)_\pi$, имеем $C_j(\pi) < C_f(\pi)$. Если $d_j \ge d_f$, то, очевидно,

$$L_j(\pi) = C_j(\pi) - d_j < C_f(\pi) - d_f = L_f(\pi),$$

поэтому справедливо (2.6).

Пусть для требования $j \in N$, $(j \to f)_\pi$, имеем $d_j < d_f$. Тогда $r_j > r_f$. Если бы $r_j \le r_f$, то и $r_j(t) \le r_f(t)$ и $r_f(t) = r(t)$, как следует из (2.1) и (2.4). Тогда $r_j(t) = r_f(t) = r(t)$ и $d_j < d_f$, но это противоречит определению требования f (2.4). Поэтому $r_j > r_f$. Очевидно, что $C_j(\pi) - p_j < C_f(\pi) - p_f$ и, так как $r_j > r_f$, то

$$C_j(\pi) - p_j - r_j < C_f(\pi) - p_f - r_f,$$

$$C_j(\pi) - C_f(\pi) < p_j + r_j - p_f - r_f.$$

Так как $d_j < d_f$, тогда из (2.3) $d_j - r_j - p_j \geq d_f - r_f - p_f$ или $d_j - d_f \geq r_j + p_j - r_f - p_f$, поэтому $C_j(\pi) - C_f(\pi) < p_j + r_j - p_f - r_f \leq d_j - d_f$. Отсюда $L_j(\pi, t) < L_f(\pi, t)$ для всех требований $j, (j \to f)_\pi$.

Аналогично докажем неравенство (2.7).

Для всех требований j, удовлетворяющих условию $(j \to s)_\pi$, имеем $C_j(\pi) < C_s(\pi)$. Если $d_j \geq d_s$, то $L_j(\pi, t) = C_j(\pi) - d_j < C_s(\pi) - d_s = L_s(\pi, t)$, поэтому справедливо (2.7).

Пусть для требования $j \in N\backslash\{f\}$, $(j \to s)_\pi$, справедливо $d_j < d_s$, тогда $r_j > r_s$. Действительно, если предположить, что $r_j \leq r_s$, то $r_j(t) \leq r_s(t)$, что следует из (2.1). Кроме того, для требования s выполнено $r_s(t) \geq r(t)$ в соответствии с определениями (2.2) и (2.5). Если $r_s(t) = r(t)$, тогда для требований j и s можно записать $r_j(t) = r_s(t) = r(t)$ и $d_j < d_s$, что противоречит определению (2.5) требования $s(N, t)$. Если же $r_s(t) > r(t)$, т.е. $r_s > r(t)$, тогда нет требования $i \in N\backslash\{f, s\}$, для которого $r_s > r_i > r(t)$. Следовательно, для требований j и s получим $r_j(t) = r_s(t)$ и $d_j < d_s$, что противоречит определению (2.5) требования $s(N, t)$. Поэтому $r_j > r_s$.

Так как $C_j(\pi) \leq C_s(\pi) - p_s$ и $p_j > 0$, то $C_j(\pi) - p_j < C_s(\pi) - p_s$ и так как $r_j > r_s$, поэтому $C_j(\pi) - p_j - r_j < C_s(\pi) - p_s - r_s$ и

$$C_j(\pi) - C_s(\pi) < p_j + r_j - p_s - r_s. \tag{2.8}$$

Поскольку $d_j < d_s$, тогда из (2.3) имеем $d_j - r_j - p_j \geq d_s - r_s - p_s$ или

$$C_j(\pi) - C_s(\pi) < p_j + r_j - p_s - r_s \leq d_j - d_s. \tag{2.9}$$

Отсюда, $L_j(\pi) < L_s(\pi)$ для всех требований $j \in N\backslash\{f\}, (j \to s)_\pi$. □

Теорема 2.1 *Если для требований подмножества $N' \subseteq N$ справедливо (2.3), то для любого момента времени $t' \geq t$ и всякого раннего расписания $\pi \in \Pi(N')$ существует $\pi' \in \Pi(N')$, что*

$$L_{\max}(\pi', t') \leq L_{\max}(\pi, t'), \quad C_{\max}(\pi', t') \leq C_{\max}(\pi, t') \tag{2.10}$$

и первым при расписании π' обслуживается требование $f = f(N', t')$ или $s = s(N', t')$. Если $d_f \leq d_s$, то первым при расписании π' обслуживается требование f.

Доказательство. Пусть $\pi = (\pi_1, f, \pi_2, s, \pi_3)$, где π_1, π_2, π_3 – частичные расписания π. Построим расписание $\pi' = (f, \pi_1, \pi_2, s, \pi_3)$. Из определений (2.1), (2.2), (2.4) получаем $r_f(t') \leq r_j(t')$, $j \in N'$, отсюда $C_{\max}((f, \pi_1), t') \leq C_{\max}((\pi_1, f), t')$ и

$$C_{\max}(\pi', t') \leq C_{\max}(\pi, t'), \tag{2.11}$$

$$L_j(\pi', t') \le L_j(\pi, t'), \ \ \forall j \in \{(\pi_2, s, \pi_3)\}. \tag{2.12}$$

Из леммы 2.1 (2.7) имеем

$$L_j(\pi', t') < L_s(\pi', t'), \ \ \forall j \in \{\pi_1\} \cup \{\pi_2\}. \tag{2.13}$$

Очевидно, что для требования f справедливо

$$L_f(\pi', t') \le L_f(\pi, t'). \tag{2.14}$$

Из (2.11)–(2.14) приходим к $C_{\max}(\pi', t') \le C_{\max}(\pi, t')$ и $L_{\max}(\pi', t') \le L_{\max}(\pi, t')$.

Пусть $\pi = (\pi_1, s, \pi_2, f, \pi_3)$, т.е. требование s обслуживается до требования f. Построим расписание $\pi' = (s, \pi_1, \pi_2, f, \pi_3)$. Далее доказательство может быть повторено как и для f. Первая часть теоремы доказана.

Предположим $d_f \le d_s$ и расписание $\pi = (\pi_1, s, \pi_2, f, \pi_3)$. Построим расписание $\pi' = (f, \pi_{11}, \pi_{12}, \pi_3)$, где π_{11}, π_{12} – расписания из требований множеств $\{j \in N' : j \in \{(\pi_1, s, \pi_2)\}, d_j < d_f\}$ и $\{j \in N' : j \in \{(\pi_1, s, \pi_2)\}, d_j \ge d_f\}$. Требования в π_{11} и π_{12} упорядочены по не убыванию моментов поступлений r_j. Из $d_s \ge d_f$ следует, что $s \in \{\pi_{12}\}$.

Для каждого требования $j \in \{\pi_{11}\}$ имеем $d_j < d_f$. Из (2.3) получаем $d_j - r_j - p_j \ge d_f - r_f - p_f$, отсюда $r_j + p_j < r_f + p_f$, $\forall j \in \{\pi_{11}\}$, и $C_{\max}((f, \pi_{11}), t') = r_f(t') + p_f + \sum\limits_{j \in \{\pi_{11}\}} p_j$. Так как требования расписания $\{\pi_{12}\}$ упорядочены по не убыванию моментов поступлений, поэтому $C_{\max}((f, \pi_{11}, \pi_{12}), t') \le C_{\max}((\pi_1, s, \pi_2, f), t')$. В результате

$$C_{\max}(\pi', t') \le C_{\max}(\pi, t'), \tag{2.15}$$

$$L_j(\pi', t') \le L_j(\pi, t'), \ \ \forall j \in \{\pi_3\}. \tag{2.16}$$

Требование $j \in \{\pi_{12}\}$ удовлетворяет $d_j \ge d_f$ и $C_j(\pi', t') \le C_f(\pi, t')$, а значит

$$L_j(\pi', t') \le L_f(\pi, t'), \ \ \forall j \in \{\pi_{12}\}. \tag{2.17}$$

Так как $s \in \{\pi_{12}\}$, то

$$L_s(\pi', t') \le L_f(\pi, t'). \tag{2.18}$$

Из леммы 2.1

$$L_j(\pi', t') \le L_s(\pi', t'), \ \ \forall j \in \{\pi_{11}\}. \tag{2.19}$$

Более того, очевидно, что

$$L_f(\pi', t') \le L_f(\pi, t'). \tag{2.20}$$

Из неравенств (2.15)–(2.20) следуют $C_{\max}(\pi', t') \leq C_{\max}(\pi, t')$ и $L_{\max}(\pi', t') \leq L_{\max}(\pi, t')$, что и требовалось доказать. □

Будем называть расписание $\pi' \in \Pi(N)$ **эффективным**, если не существует расписания $\pi \in \Pi(N)$, что $L_{\max}(\pi) \leq L_{\max}(\pi')$ и $C_{\max}(\pi) \leq C_{\max}(\pi')$, причем хотя бы одно неравенство строгое.

Таким образом, когда для требований множества N выполняются ограничения (2.3), то найдется эффективное расписание π', при котором первым обслуживается либо требование $f = f(N, t)$, либо $s = s(N, t)$. Более того, если $d_f \leq d_s$, то существует эффективное расписание π' с первоочередным обслуживанием требования f.

Определим множество расписаний $\Omega(N, t)$ как подмножество $\Pi(N)$, состоящее из $n!$ расписаний. Расписание $\pi = (i_1, i_2, \ldots, i_n)$ принадлежит $\Omega(N, t)$, если требование $i_k, k = 1, 2, ..., n$, выбирается из $f_k = f(N_{k-1}, C_{i_{k-1}})$ и $s_k = s(N_{k-1}, C_{i_{k-1}})$, где $N_{k-1} = N \setminus \{i_1, i_2, ..., i_{k-1}\}$, $C_{i_{k-1}} = C_{i_{k-1}}(\pi)$ и $N_0 = N$, $C_{i_0} = t$. Для $d_{f_k} \leq d_{s_k}$ выполняется $i_k = f_k$, если $d_{f_k} > d_{s_k}$, то $i_k = f_k$ или $i_k = s_k$. Очевидно, что множество расписаний $\Omega(N, t)$ содержит не более 2^n расписаний.

Пример 2.1

$$\begin{cases} n = 2m, t \leq r_1 < r_2 < \ldots < r_n, \\ r_{2i-1} < r_{2i} + p_{2i} < r_{2i-1} + p_{2i-1}, 1 \leq i \leq m, \\ r_{2i-1} + p_{2i-1} + p_{2i} < r_{2i+1} < r_{2i} + p_{2i} + p_{2i-1} < r_{2i+2}, 1 \leq i \leq m-1, \\ r_{2i-1} + p_{2i-1} + p_{2i} - d_{2i-1} > y, 1 \leq i \leq m-1, \\ r_{2i} + p_{2i} + p_{2i-1} - d_{2i} \leq y, \end{cases}$$

т.е. $p_{2i} > y \geq p_{2i-1}$.

Множество $\Omega(N, t)$ содержит 2^m расписаний. Величина y нами будет использоваться далее в тексте. Оптимальное решение задачи $1|r_j, d_j = r_j + p_j, L_{max} \leq y|C_{\max}$ будет $\pi^* = (2, 1, 4, 3, \ldots, n, n-1)$.

Теорема 2.2 *Если для требований подмножества $N' \subseteq N, \mid N' \mid = n'$, справедливо (2.3), то для любых момента времени $t' \geq t$ и расписания $\pi \in \Pi(N')$ существует такое расписание $\pi' \in \Omega(N', t')$, что*

$$L_{\max}(\pi', t') \leq L_{\max}(\pi, t') \;\; и \;\; C_{\max}(\pi', t') \leq C_{\max}(\pi, t').$$

Доказательство. Пусть $\pi = (j_1, j_2, \ldots, j_{n'})$ – произвольное расписание. Будем обозначать первые l требований расписания π как π_l, $l = 0, 1, 2, \ldots, n'$,

где π_0 – пустое расписание, а через $\bar{\pi}_l = (j_{l+1}, ..., j_{n'})$, тогда $\pi = (\pi_l, \bar{\pi}_l)$. Введём $N_l = N' \setminus \{\pi_l\}$ и $C_l = C_{\max}(\pi_l, t')$. Предположим, что для некоторого $l, 0 \leq l < n'$, π_l является наибольшим начальным частичным расписанием некоторого расписания из $\Omega(N', t')$. Если $j_1 \neq f(N', t')$ и $j_1 \neq s(N', t')$, то $\pi_l = \pi_0$, $l = 0$, т.е. наибольшим частичным расписанием будет пустое. Допустим $f = f(N_l, C_l)$ и $s = s(N_l, C_l)$. Если $d_f > d_s$, то $j_{l+1} \neq f$ и $j_{l+1} \neq s$, более того когда $d_f \leq d_s$, то $j_{l+1} \neq f$, так как π_{l+1} не является начальным расписанием некоторого расписания из $\Omega(N', t')$.

Согласно теореме 2.1 для требований множества $\{\bar{\pi}_l\}, \bar{\pi}_l \in \Pi(N_l)$, с момента времени C_l существует расписание $\bar{\pi}'_l$, для которого выполняется $L_{\max}(\bar{\pi}'_l, C_l) \leq L_{\max}(\bar{\pi}_l, C_l)$ и $C_{\max}(\bar{\pi}'_l, C_l) \leq C_{\max}(\bar{\pi}_l, C_l)$, и $[\bar{\pi}'_l]_1 = f$ или s, более того, при $d_f \leq d_s$, справедливо $[\bar{\pi}'_l]_1 = f$, где $[\sigma]_k$ – требование на k-том месте при расписании σ. Отсюда, $L_{\max}((\pi_l, \bar{\pi}'_l), t') \leq L_{\max}((\pi_l, \bar{\pi}_l), t')$ и $C_{\max}((\pi_l, \bar{\pi}'_l), t') \leq C_{\max}((\pi_l, \bar{\pi}_l), t')$.

Обозначим $\pi' = (\pi_l, \bar{\pi}'_l)$. Мы получили расписание π', отличающееся тем, что обслуживание первых $l+1$ требований будет совпадать с обслуживанием первых $l+1$ требований некоторого расписания из множества $\Omega(N', t')$ и $L_{\max}(\pi', t') \leq L_{\max}(\pi, t')$, $C_{\max}(\pi', t') \leq C_{\max}(\pi, t')$.

После не более чем n' последовательных преобразований (так как длина расписания $n' \leq n$) исходного произвольно выбранного расписания π мы приходим к расписанию $\pi' \in \Omega(N', t')$, обеспечивающему $L_{\max}(\pi', t') \leq L_{\max}(\pi, t')$ и $C_{\max}(\pi', t') \leq C_{\max}(\pi, t')$, что и требовалось доказать. □

Сформируем $\omega(N, t) = (i_1, i_2, \ldots, i_l)$ следующее частичное расписание. Для каждого требования i_k, $k = 1, 2, ..., l$, имеем $i_k = f_k$ и $d_{f_k} \leq d_{s_k}$, где $f_k = f(N_{k-1}, C_{k-1})$ и $s_k = s(N_{k-1}, C_{k-1})$. Для $f = f(N_l, C_l)$ и $s = s(N_l, C_l)$ выполняется $d_f > d_s$. В случае, когда $d_f > d_s$ для $f = f(N, t)$ и $s = s(N, t)$, то $\omega(N, t) = \emptyset$. Таким образом, $\omega(N, t)$ – "максимальное" расписание, при построении которого однозначно выбирается требование (типа f) на очередное место расписания. С помощью алгоритма 2.1 для требований множества N с момента времени t может быть построено расписание $\omega(N, t)$.

Лемма 2.2 *Трудоёмкость алгоритма 2.1 нахождения расписания $\omega(N, t)$ для любых N и t составляет не более $O(n \log n)$ операций.*

Доказательство. На каждой итерации алгоритма 2.1 находим два требования: $f = f(N, t)$ и $s = s(N, t)$. Если требования упорядочены по временам поступления r_j (и, соответственно, момент времени $r(t)$ находится за $O(1)$ операций), то для нахождения двух требований (f и s) необходимо $O(\log n)$

Алгоритм 2.1 Алгоритм построения $\omega(N,t)$

1: **Вспомогательный шаг.** Положим $\omega = \emptyset$.
2: **Основной шаг.** Найдем требования $f := f(N,t)$ и $s := s(N,t)$;
3: **if** $d_f \le d_s$ **then**
4: $\omega := (\omega, f)$
5: **else**
6: алгоритм заканчивает работу;
7: **end if**
8: положим $N := N \setminus \{f\}$, $t := r_f(t) + p_f$ и переходим к выполнению следующего шага, начиная с основного.

операций. Всего итераций не более n. Таким образом, для построения расписания $\omega(N,t)$ требуется $O(n \log n)$ операций. □

Так как ключевым моментом в алгоритме 2.1 является нахождение требований f и s на каждом основном шаге алгоритма и, как известно, для нахождения элемента в множестве требуется не менее $O(\log n)$ операций в общем случае. Очевидно, что количество итераций алгоритма будет O(n), поэтому трудоёмкость алгоритма 2.1 построения расписания $\omega(N,t)$ за $O(n \log n)$ операций будет минимально возможной.

Лемма 2.3 *Если для требований множества N справедливо (2.3), то любое расписание $\pi \in \Omega(N,t)$ начинается с расписания $\omega(N,t)$.*

Доказательство. Когда $\omega(N,t) = \emptyset$, т.е. $d_f > d_s$, где $f = f(N,t)$, $s = s(N,t)$, утверждение леммы справедливо, так как любое расписание начинается с пустого.

Пусть $\omega(N,t) = (i_1, i_2, \ldots, i_l)$, $l > 0$, а значит для каждого i_k, $k = 1, 2, \ldots, l$, имеем $i_k = f_k$ и $d_{f_k} \le d_{s_k}$, где $f_k = f(N_{k-1}, C_{k-1})$ и $s_k = s(N_{k-1}, C_{k-1})$. Для $f = f(N_l, C_l)$ и $s = s(N_l, C_l)$ справедливо $d_f > d_s$. Как видно из определения множества расписаний $\Omega(N,t)$ все расписания этого подмножества будут начинаться с частичного расписания $\omega(N,t)$. □

Воспользуемся следующими обозначениями $\omega^1(N,t) = (f, \omega(N', t'))$ и $\omega^2(N,t) = (s, \omega(N'', t''))$, где $f = f(N,t)$, $s = s(N,t)$, $N' = N \setminus \{f\}$, $N'' = N \setminus \{s\}$, $t' = r_f(t) + p_f$, $t'' = r_s(t) + p_s$. Очевидно, что алгоритм для нахождения ω^1 (а также ω^2) выполняется за $O(n \log n)$ операций также, как и алгоритм для построения $\omega(N,t)$.

Следствие 2.1 из леммы 2.3. *Если для требований множества N справедливо (2.3), то каждое расписание $\pi \in \Omega(N,t)$ начинается или с $\omega^1(N,t)$, или с $\omega^2(N,t)$.* □

Теорема 2.3 *Если требования множества N удовлетворяют (2.3), то для любого расписания $\pi \in \Omega(N,t)$ выполняется $(i \to j)_\pi$ для каждого $i \in \{\omega^1(N,t)\}$ и $j \in N \setminus \{\omega^1(N,t)\}$.*

Доказательство. В случае $\{\omega^1(N,t)\} = N$ утверждение теоремы, очевидно, справедливо. Пусть $\{\omega^1(N,t)\} \neq N$. Далее в тексте доказательства будем использовать обозначение $\omega^1 = \omega^1(N,t)$.

Если $f = f(N,t)$ и $s = s(N,t)$ таковы, что $d_f \leq d_s$, то все расписания из множества $\Omega(N,t)$ начинаются с частичного расписания $\omega(N,t) = \omega^1$, поэтому утверждение теоремы также верно.

Рассмотрим случай $d_f > d_s$. Все расписания множества $\Omega(N,t)$, начинающиеся с требования f, имеют частичное расписание $\omega(N,t) = \omega^1$.

Возьмём произвольное расписание $\pi \in \Omega(N,t)$ с требованием s на первом месте, $[\pi]_1 = s$, и расписание $\mid \omega^1 \mid = l, l < n$, содержащее l требований. Пусть $\pi_l = (j_1, j_2, \ldots, j_l)$ – частичное расписание длины l расписания π, $j_1 = s$. Докажем, что $\{\pi_l\} = \{\omega^1\}$. Предположим противное, что существует требование $j \in \{\pi_l\}$, но $j \notin \{\omega^1\}$.

Предположим, что $(j \to f)_\pi$. Если $d_j < d_f$, то из (2.3) имеем $d_j - r_j - p_j \geq d_f - r_f - p_f$, поэтому $r_j + p_j < r_f + p_f$. Тогда требование j будет включено в расписание ω^1 согласно определению $\omega(N,t)$ и ω^1, но по предположению $j \notin \{\omega^1\}$. Если же $d_j \geq d_f$, то из того, что $\pi \in \Omega(N,t)$ следует $(f \to j)_\pi$, но это противоречит $(j \to f)_\pi$. Следовательно $j \in \{\omega^1\}$.

Пусть $(f \to j)_\pi$. Тогда для каждого требования $i \in \{\omega^1\}$, для которого $i \notin \{\pi_l\}$, $r_i < r_i + p_i \leq C_{\max}(\omega^1) < r_{s_{l+1}} \leq r_j$, т.к. $j \notin \{\omega^1\}$, где $s_{l+1} = s(N \setminus \{\omega^1\}, C_{\max}(\omega^1))$. Требования s_{l+1} и j не были упорядочены при расписании ω^1, поэтому $C_{\max}(\omega^1) < r_{s_{l+1}} \leq r_j$. Кроме того, $d_i \leq d_j$. Если бы $d_i > d_j$, то $r_i + p_i \geq r_j + p_j$, но выполняется $r_i + p_i < r_j$. Следовательно $(i \to j)_{\pi_l}$, так как $\pi = (\pi_l, \bar{\pi}_l) \in \Omega(N,t)$, но тогда будет нарушено предположение $i \notin \{\pi_l\}$ и $j \in \{\pi_l\}$.

Следовательно наше предположение не верно, поэтому $\{\omega^1\} = \{\pi_l\}$, что и требовалось доказать. □

Таким образом, обслуживание требований множества $\{\omega^1(N,t)\}$ предшествует обслуживанию требований множества $N \setminus \{\omega^1(N,t)\}$ при любом расписании из множества $\Omega(N,t)$, среди которых находится и оптимальное расписание.

2.1.2 Задача на быстродействие при ограничении на максимальное временно́е смещение

Сформулируем задачу $1 \mid d_i \le d_j, d_i - r_i - p_i \ge d_j - r_j - p_j; L_{\max} \le y \mid C_{\max}$ – необходимо для некоторого действительного числа y найти расписание θ с $C_{\max}(\theta) = \min\{C_{\max}(\pi) : L_{\max}(\pi) \le y\}$. Если $L_{\max}(\pi) > y$ для любого $\pi \in \Pi(N)$, то $\theta = \emptyset$.

Алгоритм 2.2 решения задачи $1 \mid d_i \le d_j, d_i - r_i - p_i \ge d_j - r_j - p_j; L_{\max} \le y \mid C_{\max}$

Вспомогательный шаг. Положим $\theta := \omega(N, t), t' := t$;
Основной шаг.
if $L_{\max}(\theta, t') > y$ **then**
  $\theta := \emptyset$ и алгоритм заканчивает работу.
end if
Найдём $N' := N \setminus \{\theta\}$, $t' := C_{\max}(\theta)$ и $\omega^1(N', t'),\ \omega^2(N', t')$.
if $N' = \emptyset$ **then**
  выполнение алгоритма прекращается.
else
  if $L_{\max}(\omega^1, t') \le y$ **then**
    $\theta := (\theta, \omega^1)$ и переходим к следующему шагу;
  end if
  if $L_{\max}(\omega^1, t') > y$ и $L_{\max}(\omega^2, t') \le y$ **then**
    $\theta := (\theta, \omega^2)$ и переходим к следующему шагу;
  end if
  if $L_{\max}(\omega^1, t') > y$ и $L_{\max}(\omega^2, t') > y$ **then**
    $\theta := \emptyset$ и алгоритм заканчивает работу.
  end if
end if

Лемма 2.4 *Трудоёмкость алгоритма 2.2 не превышает* $O(n^2 \log n)$ *операций.*

Доказательство. На каждой итерации основного шага алгоритма 2.2 находятся расписания ω^1 и, при необходимости, ω^2 за $O(n \log n)$ операций. Так как ω^1 и ω^2 состоят по крайней мере из одного требования, то на каждой итерации алгоритма к расписанию θ добавляется одно или несколько требований, или полагаем $\theta = \emptyset$ и завершаем работу. Поэтому общее число шагов алгоритма не более n. Таким образом, алгоритм 2.2 выполняется за $O(n^2 \log n)$ операций. □

Решить задачу $1 \mid d_i \le d_j, d_i - r_i - p_i \ge d_j - r_j - p_j; L_{\max} \le y \mid C_{\max}$ за меньшее число операций, чем $O(n^2 \log n)$, невозможно, так как существует пример 2.1, на котором достигается такая сложность. Оптимальным расписанием будет $\pi^* = (2, 1, 4, 3, \ldots, n, n-1)$. Для его нахождения требуется $O(n^2 \log n)$ операций.

Также, как и для алгоритма 2.1, трудоёмкость которого $O(n \log n)$ операций, алгоритм 2.2 имеет нижнюю границу алгоритмической сложности решения задачи $1|d_i \leq d_j, d_i - r_i - p_i \geq d_j - r_j - p_j; L_{\max} \leq y|C_{\max}$, равную $O(n^2 \log n)$ операций. Будем обозначать через $\theta(N, t, y)$ расписание, построенное алгоритмом 2.2 с момента времени t из требований множества N с максимальным временны́м смещением не больше y. Если $N = \emptyset$, то $\theta(\emptyset, t, y) = \emptyset$ для любого t и y.

Теорема 2.4 *Пусть для требований множества N справедливо (2.3). Если с помощью алгоритма 2.2 будет построено расписание $\theta(N, t, y) \neq \emptyset$, то $C_{\max}(\theta) = \min\{C_{\max}(\pi) : L_{\max}(\pi) \leq y, \pi \in \Pi(N)\}$. Если же в результате работы алгоритма 2.2 не будет сформировано расписание, т.е. $\theta(N, t, y) = \emptyset$, то $L_{\max}(\pi) > y$ для каждого $\pi \in \Pi(N)$.*

Доказательство. В ситуации, когда для расписания $\pi \in \Pi(N)$ выполняется $L_{\max}(\pi) \leq y$, существует такое расписание $\pi' \in \Omega(N, t)$, что $L_{\max}(\pi') \leq L_{\max}(\pi) \leq y$ и $C_{\max}(\pi') \leq C_{\max}(\pi)$,– согласно теореме 2.2. Поэтому искомое расписание θ находится среди расписаний множества $\Omega(N, t)$.

По лемме 2.3, все расписания множества $\Omega(N, t)$ начинаются с $\omega(N, t)$. Примем $\theta_0 = \omega(N, t)$.

После $k, k \geq 0$, основных шагов алгоритма 2.2 мы получили расписание θ_k и $N' = N \setminus \{\theta_k\}$, $t' = C_{\max}(\theta_k)$. Предположим, что существует оптимальное по быстродействию расписание θ, начинающееся с θ_k. Согласно теореме 2.2 возможно оптимальное продолжение расписания θ_k среди расписаний из множества $\Omega(N', t')$.

Пусть $\theta_{k+1} = (\theta_k, \omega^1(N', t'))$, т.е. $L_{\max}(\theta_{k+1}) \leq y$. При расписании ω^1, $\omega^1 = \omega^1(N', t')$, нет искусственных простоев прибора и все расписания из множества $\Omega(N', t')$ начинаются с требований множества $\{\omega^1(N', t')\}$,– по теореме 2.3. Поэтому $\omega^1(N', t')$ наилучшее по быстродействию $(C_{\max})$ среди всех допустимых по максимальному временно́му смещению $(L_{\max})$ продолжений частичного расписания θ_k.

Если $\theta_{k+1} = (\theta_k, \omega^2(N', t'))$, т.е. $L_{\max}(\omega^1, t') > y$, и $L_{\max}(\omega^2, t') \leq y$. Все расписания множества $\Omega(N', t')$ начинаются с расписания $\omega^1(N', t')$ или с $\omega^2(N', t')$. Так как $L_{\max}(\omega^1, t') > y$, то существует только одно подходящее продолжение – $\omega^2(N', t')$.

Таким образом, на каждом основном шаге алгоритма мы выбираем наилучшее по быстродействию продолжение частичного расписания θ_k среди всех допустимых по максимальному временно́му смещению. После не более чем n основных шагов алгоритма будет построено искомое расписание.

Допустим, что после $k+1$ шага алгоритма $L_{\max}(\omega^1, t') > y$ и $L_{\max}(\omega^2, t') > y$. Если расписание θ существовало бы, т.е. $\theta \neq \emptyset$, то θ начиналось бы с θ_k. Тогда для любого расписания $\pi \in \Pi(N', t')$ существует такое расписание $\pi' \in \Omega(N', t')$, что $L_{\max}(\pi, t') \geq L_{\max}(\pi', t') \geq L_{\max}(\omega^1, t') > y$ или $L_{\max}(\pi, t') \geq L_{\max}(\pi', t') \geq L_{\max}(\omega^2, t') > y$. Поэтому $\theta = \emptyset$.

Повторив наши рассуждения столько раз, сколько выполнялся основной шаг алгоритма 2.2 (не более чем n), мы придем к истинности утверждения теоремы. □

2.1.3 Алгоритм построения множества Парето-расписаний по критериям $C_{\max}$ и $L_{\max}$

Ниже приводится алгоритм построения множества Парето-расписаний $\Phi(N, t) = \{\pi'_1, \pi'_2, \ldots, \pi'_m\}$, $m \leq n$, по критериям $C_{\max}$ и $L_{\max}$ таких, что

$$C_{\max}(\pi'_1) < C_{\max}(\pi'_2) < \ldots < C_{\max}(\pi'_m),$$
$$L_{\max}(\pi'_1) > L_{\max}(\pi'_2) > \ldots > L_{\max}(\pi'_m).$$

Расписание π'_m будет решением задачи $1 \mid r_j \mid L_{\max}$ при условии, что выполняется (2.3). В результате работы алгоритма 2.3 для множества требований N с момента времени t будет построено множество расписаний $\Phi(N, t)$, для которого верно $1 \leq \mid \Phi(N, t) \mid \leq n$. Заметим, что множество $\Phi(N, t)$ примера 2.1 будет состоять из двух расписаний, хотя множество $\Omega(N, t)$ содержит $2^{\frac{n}{2}}$ расписаний:

$$\pi'_1 = (1, 2, 3, 4, \ldots, n-1, n);$$
$$\pi'_2 = (2, 1, 4, 3, \ldots, n, n-1).$$

Лемма 2.5 *Трудоёмкость алгоритма 2.3 не превышает* $O(n^3 \log n)$ *операций.*

Доказательство. На каждой итерации основного шага алгоритма 2.3 находятся расписания ω^1 и, при необходимости, ω^2 за $O(n \log n)$ операций согласно лемме 2.2, а также расписание θ за $O(n^2 \log n)$ операций. Так как ω^1 и ω^2 состоят по крайней мере из одного требования, то на любой итерации алгоритма к расписанию π^* добавляется одно или больше требований или происходит остановка на последнем эталонном расписании π'. Поэтому общее число итераций не более n. Таким образом, алгоритм 2.3 выполняется не более чем за $O(n^3 \log n)$ операций. □

Для примера 2.1 в случае, когда

$$p_{2n-1} < p_{2n-3} < \ldots < p_1 \leq y < p_2 < p_4 < \ldots < p_2 n$$

решения π'_1 и π'_2 будут найдены за $O(n^3 \log n)$ операций — это верхняя достижимая граница трудоемкости алгоритма 2.3.

Алгоритм 2.3 построения множества Парето-расписаний по критериям $C_{\max}$ и $L_{\max}$

Вспомогательный шаг. $y := +\infty$, $\pi^* := \omega(N, t)$, $\Phi := \emptyset$, $m := 0$, $N' := N \setminus \{\pi^*\}$ и $t' := C_{\max}(\pi^*)$.
if $N' = \emptyset$ **then**
  $\Phi := \Phi \cup (\pi^*), m := 1$ и алгоритм заканчивает работу.
end if
Основной шаг.
if $L_{\max}(\omega^1, t') \leq L_{\max}(\pi^*)$ **then**
  $\pi^* := (\pi^*, \omega^1)$, где $\omega^1 = \omega^1(N', t')$ и переходим к следующему шагу;
end if
if $L_{\max}(\omega^1, t') > L_{\max}(\pi^*)$ **then**
  if $L_{\max}(\omega^1, t') < y$ **then**
    находим $\theta = \theta(N', t', y')$ с помощью алгоритма 2.2, где $y' = L_{\max}(\omega^1, t')$;
    if $\theta = \emptyset$ **then**
      $\pi^* := (\pi^*, \omega^1)$ и переходим к следующему шагу;
    else
      $\pi' := (\pi^*, \theta)$
      if $C_{\max}(\pi'_m) < C_{\max}(\pi')$ **then**
        $m := m + 1$, $\pi'_m := \pi'$, $\Phi := \Phi \cup (\pi'_m)$, $y = L_{\max}(\pi'_m)$;
      else
        $\pi'_m = \pi'$ и переходим к следующему шагу;
      end if
    end if
    if $L_{\max}(\omega^1, t') \geq y$ **then**
      находим $\omega^2 = \omega^2(N', t')$;
      if $L_{\max}(\omega^2, t') < y$ **then**
        $\pi^* = (\pi^*, \omega^2)$ и переходим к следующему шагу;
      else
        $\pi^* = \pi'_m$ и алгоритм заканчивает работу.
      end if
    end if
  end if
end if

Также как алгоритмы 2.1 и 2.2, алгоритм 2.3 имеет минимальную алгоритмическую сложность ($O(n^3 \log n)$ операций) решения задачи построения Парето-множества расписаний по критериям $C_{\max}$ и $L_{\max}$.

Теорема 2.5 *Если для требований множества N справедливо (2.3), тогда расписание π'_m, построенное алгоритмом 2.3, является оптимальным по критерию $L_{\max}$. Кроме того, для любого расписания $\pi \in \Pi(N)$ имеется расписание $\pi' \in \Phi(N,t)$, что $L_{\max}(\pi') \le L_{\max}(\pi)$ и $C_{\max}(\pi') \le C_{\max}(\pi)$ и $\mid \Phi(N,t) \mid \le n$.*

Доказательство. Согласно теореме 2.2 существует оптимальное (по $L_{\max}$) расписание, принадлежащее множеству $\Omega(N,t)$. Все расписания множества $\Omega(N,t)$ начинаются с частичного расписания $\omega(N,t)$, – согласно лемме 2.3.

Пусть $\pi_0 = \omega(N,t)$. После k, $k \ge 0$, основных шагов алгоритма 2.3 мы имеем частичное расписание π_k. Допустим наличие оптимального (по $L_{\max}$) расписания начинающегося с π_k. Обозначим $N' = N \setminus \{\pi_k\}$ и $t' = C_{\max}(\pi_k)$.

Если $\pi_{k+1} = (\pi_k, \omega^1)$, где $\omega^1 = \omega^1(N't')$, то или $L_{\max}(\omega^1, t') \le L_{\max}(\pi_k)$, или $L_{\max}(\pi_k) < L_{\max}(\omega^1, t') < y$ – текущее значение критерия и максимальное временно́е смещение "появится" на следующих шагах алгоритма 2.3, т.е. $\theta(N', t', y') = \emptyset$, где $y' = L_{\max}(\omega^1, t')$. Если же $\theta = \theta(N', t', y') \ne \emptyset$, то мы улучшаем текущее значение максимального временно́го смещения: $\pi' = (\pi_k, \theta)$ и $y = L_{\max}(\pi') = L_{\max}(\omega^1, t')$. Расписание π' добавляется в множество расписаний $\Phi(N,t)$. Более того, обслуживание требований множества $\{\omega^1\}$ предшествуют обслуживанию требований множества $N' \setminus \{\omega^1\}$, – согласно теореме 2.3. Таким образом, расписание ω^1 без искусственных простоев прибора будет лучшим продолжением для π_k.

Рассмотрим случай $\pi_{k+1} = (\pi_k, \omega^2)$, где $\omega^2 = \omega^2(N', t')$, т.е. $L_{\max}(\omega^2, t') < L_{\max}(\pi') \le L_{\max}(\omega^1, t')$, – согласно алгоритму 2.3. Поэтому продолжение ω^2 "лучше", чем ω^1. Отсюда частичное расписание π_{k+1} является частью некоторого оптимального расписания.

Повторяя подобные рассуждения не более n раз, мы приходим к оптимальности (по $L_{\max}$) расписания π^*.

Множество расписаний $\Phi(N,t)$ будет содержать не более n расписаний, так как на каждом основном шаге алгоритма в множество $\Phi(N,t)$ "добавляется" не более одного расписания, а этот шаг выполняется не более n раз.

Предположим, что существует расписание $\pi \in \Pi(N), \pi \notin \Phi(N,t)$, такое, что либо $C_{\max}(\pi) \le C_{\max}(\pi')$ и $L_{\max}(\pi) \ge L_{\max}(\pi')$, либо $C_{\max}(\pi) \ge C_{\max}(\pi')$ и $L_{\max}(\pi) \le L_{\max}(\pi')$ для каждого расписания $\pi' \in \Phi(N,t)$, причем в каждой паре неравенств хотя бы одно из неравенств строгое. Из теоремы 2.1 следует наличие расписания $\pi'' \in \Omega(N,t)$ такого, что $L_{\max}(\pi'') \le L_{\max}(\pi)$ и $C_{\max}(\pi'') \le C_{\max}(\pi)$. Если $\pi'' \in \Phi(N,t)$, то, очевидно, что наше предположение не верно. Пусть $\pi'' \in \Omega(N,t) \setminus \Phi(N,t)$. Из алгоритма 2.3 видно,

что структуру каждого расписания $\pi' \in \Phi(N,t)$ можно представить как последовательность частичных расписаний $\pi' = (\omega'_0, \omega'_1, \omega'_2, \ldots, \omega'_{k'})$, где $\omega'_0 = \omega(N,t), \omega'_i$ равно либо $\omega^1(N'_i, C'_i)$, либо $\omega^2(N'_i, C'_i)$, а $N'_i = N \backslash \{\omega'_0, \ldots, \omega'_{i-1}\}$, $C'_i = C_{\max}((\omega'_0, \ldots, \omega'_{i-1}), t), i = 1, 2, \ldots, k'$. Расписание π'' будет обладать аналогичной структурой, согласно определению множества $\Omega(N,t)$, т.е. $\pi = (\omega''_0, \omega''_1, \omega''_2, \ldots, \omega''_{k''})$, возможно, $k'' \neq k'$, где $\omega''_0 = \omega'_0 = \omega(N,t)$, ω''_i равно либо $\omega^1(N''_i, C''_i)$, либо $\omega^2(N''_i, C''_i)$, а $N''_i = N \backslash \{\omega''_0, \ldots, \omega''_{i-1}\}$, $C''_i = C_{\max}((\omega''_0, \ldots, \omega''_{i-1}), t), i = 1, 2, \ldots, k''$.

Примем, что первые k частичных расписаний π'' и π' совпадают, т.е. $\omega''_i = \omega'_i = \omega_i, i = 0, 1, \ldots, k-1, \omega''_k \neq \omega'_k$. Пусть $y = L_{\max}(\omega_0, \ldots, \omega_{k-1})$, построим расписание θ с помощью алгоритма 2.2, $\theta = \theta(N_k, C_k, y)$. Если $\theta = \emptyset$, то в соответствии с алгоритмом 2.3, $\omega'_k = \omega^1(N_k, C_k)$, так как $\omega''_k \neq \omega'_k$, то $\omega''_k = \omega^2(N_k, C_k)$. Значение целевой функции ($L_{\max}$) будет достигаться на требовании из множества N_k, так как $\theta = \emptyset$. Вся структура алгоритма 2.3 построена таким образом, чтобы до "критического" требования (по $L_{\max}$) упорядочить требования как можно более "плотно", поэтому достраиваем расписание ω^1, после чего $C_{\max}(\pi') \leq C_{\max}(\pi'')$ и $L_{\max}(\pi') \leq L_{\max}(\pi'')$. Если $\theta \neq \emptyset$, тогда для расписаний π' и π'' справедливо $C_{\max}(\pi') \leq C_{\max}(\pi'')$ и $L_{\max}(\pi') = L_{\max}(\pi'')$. Таким образом, для любого расписания $\pi'' \in \Omega(N,t) \backslash \Phi(N,t)$ существует расписание $\pi' \in \Phi(N,t)$, что $C_{\max}(\pi') \leq C_{\max}(\pi'')$ и $L_{\max}(\pi') \leq L_{\max}(\pi'')$. Из этого следует, что для любого расписания $\pi \in \Pi(N)$ существует расписание $\pi' \in \Phi(N,t)$, что $L_{\max}(\pi') \leq L_{\max}(\pi)$ и $C_{\max}(\pi') \leq C_{\max}(\pi)$. □

На рисунке 2.1 схематично показаны рассмотренные расписания.

Для множества расписаний $\Phi(N,t) = \{\pi'_1, \pi'_2, \ldots, \pi'_m\}$, $m \leq n$, имеем

$$C_{\max}(\pi'_1) < C_{\max}(\pi'_2) < \ldots < C_{\max}(\pi'_m),$$

$$L_{\max}(\pi'_1) > L_{\max}(\pi'_2) > \ldots > L_{\max}(\pi'_m).$$

Расписание π'_1 является оптимальным по быстродействию (по $C_{\max}$), а π'_m – по максимальному временно́му смещению (по $L_{\max}$), если параметры требований множества N удовлетворяют условиям (2.3).

Экспериментальное исследование алгоритма 2.3 показало, что с его помощью может быть построены оптимальные расписания (по $L_{\max}$) и для примеров, не удовлетворяющих условиям (2.3).

2.2 "Двойственная" задача

Рассмотрим общую постановку NP-трудной задачи минимизации максимального штрафа $1 \mid r_j \mid \varphi_{\max}$. Как и ранее, прибор не может обслуживать более

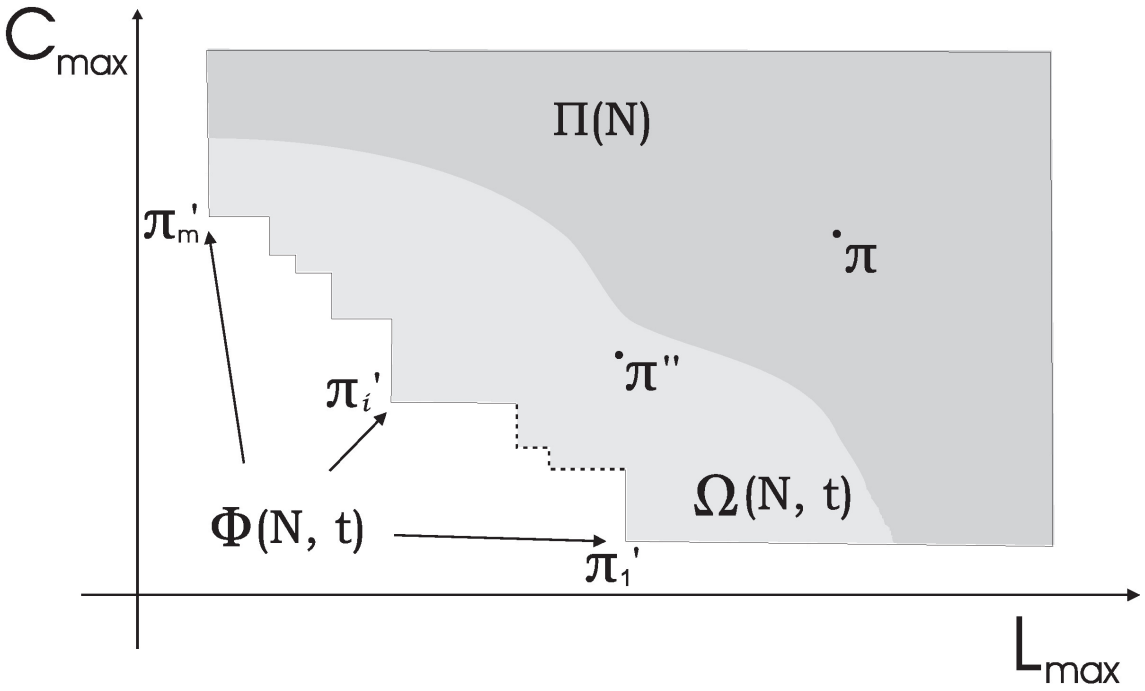

Рис. 2.1: Множество Парето-оптимальных расписаний

одного требования в любой момент времени и запрещены прерывания при обслуживании требований. Через μ^* будем обозначать оптимальное значение целевой функции:

$$\mu^* = \min_{\pi \in \Pi(N)} \max_{k=\overline{1,n}} \varphi_{j_k}(C_{j_k}(\pi)), \tag{2.21}$$

где $\varphi_{j_k}(C_{j_k}(\pi))$ – произвольные неубывающие функции штрафа, $k = \overline{1,n}$.

Сформулируем следующую задачу. Необходимо найти

$$\nu^* = \max_{k=\overline{1,n}} \min_{\pi \in \Pi(N)} \varphi_{j_k}(C_{j_k}(\pi)). \tag{2.22}$$

Для удобства введём обозначения, учитывающие место требования в расписании. Пусть расписание $\pi \in \Pi(N)$ имеет вид $\pi = (j_1, j_2, \ldots, j_n)$. Для требования, обслуживаемого k-ым, $k = 1, 2, \ldots, n$, по порядку при расписании π будем обозначать:

$$\nu_k = \min_{\pi \in \Pi(N)} \varphi_{j_k}(C_{j_k}(\pi)), k = 1, 2, \ldots, n. \tag{2.23}$$

Очевидно,

$$\nu^* = \max_{k=\overline{1,n}} \nu_k. \tag{2.24}$$

Лемма 2.6 *[19, 24] Пусть все $\varphi_j(t), j = 1, 2, \ldots, n$, произвольные неубывающие функции штрафа задачи $1 \mid r_j \mid \varphi_{\max}$, тогда для всех $k = 1, 2, \ldots, n$ выполняется $\nu_n \geq \nu_k$, т.е. $\nu^* = \nu_n$.*

Доказательство. Предположим, существует $k, k < n$, что $\nu_k > \nu_n$. Пусть $\pi_n = (j_1, j_2, \ldots, j_n)$ – расписание, на котором достигается значение ν_n. Построим расписание $\pi = (j_{n-k+1}, j_{n-k+2}, \ldots, j_n, j_1, j_2, \ldots, j_{n-k})$. При расписании π требование j_n обслуживается k-ым по порядку. Поскольку, очевидно, $C_{j_n}(\pi_n) \geq C_{j_n}(\pi)$, то имеем $\nu_n = \varphi_{j_n}(C_{j_n}(\pi_n)) \geq \varphi_{j_n}(C_{j_n}(\pi)) \geq \nu_k$. Так как по предположению $\nu_k > \nu_n$, то получаем противоречивое неравенство $\nu_k > \nu_n \geq \nu_k$. Лемма доказана. □

Таким образом, решение "двойственной" задачи сводится к нахождению ν_n. Так как $\nu_n = \min\limits_{\pi \in \Pi(N)} \varphi_{j_n}(C_{j_n}(\pi))$, то на последнее ($n$-е) место поочерёдно будем ставить каждое из требований j множества N. Оставшиеся $(n-1)$ требований множества $N \setminus \{j\}$ упорядочим в порядке поступления на обслуживание, – такой порядок даёт наименьшее значение момента окончания обслуживания требований множества $N \setminus \{j\}$.

Алгоритм 2.4 Алгоритм решения двойственной задачи к задаче $1 \mid r_j \mid \varphi_{\max}$

Построим расписание $\pi^r = (i_1, i_2, \ldots, i_n)$, при котором требования упорядочены в порядке неубывания моментов поступления: $r_{i_1} \leq r_{i_2} \leq \ldots \leq r_{i_n}$.
Для расписания $\pi_k = (\pi^r \setminus i_k, i_k), k = 1, 2, \ldots, n$, вычислим $\varphi_{i_k}(C_{i_k}(\pi_k))$.
Найдём $\nu^* = \max\limits_{k=\overline{1,n}} \varphi_{i_k}(C_{i_k}(\pi_k))$ и номер требования i_k, на котором достигается значение ν^*.

Для построения расписания π^r требуется $O(n \log n)$ операций. Для нахождения $\varphi_{i_k}(C_{i_k}(\pi_k))$ необходимо $O(n)$ операций. Таких значений n. Следовательно, для нахождения ν^* и соответствующего требования i_k, на котором достигается ν^*, потребуется не более $O(n^2)$ операций.

Теорема 2.6 *Пусть все $\varphi_j(t), j = 1, 2, \ldots, n$, произвольные неубывающие функции штрафа задачи $1 \mid r_j \mid \varphi_{\max}$, тогда $\mu^* \geq \nu^*$.*

Доказательство. Предположим противное, т.е. существует пример задачи $1 \mid r_j \mid \varphi_{\max}$, для которого $\mu^* < \nu^*$. Пусть $\pi^* = (j_1, j_2, \ldots, j_n)$ оптимальное расписание данного примера, тогда $\varphi_{j_n}(C_{j_n}(\pi^*)) \leq \mu^* < \nu^*$, что противоречит определению (2.23) $\nu^* = \nu_n = \min\limits_{\pi \in \Pi(N)} \varphi_{j_n}(C_{j_n}(\pi))$. Теорема доказана. □

Полученная оценка может быть эффективно использована при построении схем метода ветвей и границ решения задачи $1 \mid r_j \mid \varphi_{\max}$ и для оценки погрешности приближённых решений.

В алгоритме 2.5 реализована одна из возможных схем метода ветвей и границ с использованием решения "двойственной" задачи. Другие схемы метода ветвей и границ для решения задачи минимизации максимального

временно́го смещения можно найти, например, в [4, 14, 64, 93, 101, 173]. Ветвление в алгоритме 2.5 осуществляется в результате разбиения очередного подпримера на два: на очередное место расписания ставим требование f (требование с наименьшим директивным сроком из числа готовых к обслуживанию требований) и запрещаем включение требования f на очередное место расписания (за счёт увеличения возможного момента начала обслуживания требования f).

Будем обозначать через $\{N', \tau', \nu', \pi'\}$ пример построения оптимального расписания обслуживания требований множества $N' \subseteq N$ с момента времени $\tau' \geq \tau$, где ν' – оценка снизу, полученная при решении двойственной задачи данного примера, а π' – некоторое расписание обслуживания требований множества $N \setminus N', \tau' = C_{\max}(\pi', \tau), \tau$ – момент времени, с которого прибор готов начать обслуживать требования множества N.

Алгоритм 2.5 Алгоритм решения задачи $1 \mid r_j \mid L_{\max}$ методом ветвей и границ на основе решения двойственной задачи

Первоначально полагаем $\pi^* = \emptyset$. В список примеров включаем исходный пример $\{N, \tau, \nu, \emptyset\}$, где ν – решение "двойственной" задачи данного примера.
На основном шаге алгоритма делаем следующее. Из списка примеров выбираем пример с минимальным значением оценки снизу (в другом варианте алгоритма – пример с наименьшим количеством неупорядоченных требований) – $\{N', \tau', \nu', \pi'\}$. Найдём требование $f = f(N', \tau')$ (требование из множества N' с наименьшим директивным сроком из числа готовых к обслуживанию с момента времени τ' требований). В список примеров вместо примера $\{N', \tau', \nu', \pi'\}$ включаем два подпримера: $\{N_1, \tau_1, \nu_1, \pi_1\}$ и $\{N_2, \tau_2, \nu_2, \pi_2\}$, где $N_1 = N' \setminus \{f\}, \tau_1 = \max\{r_f, \tau'\} + p_f, \nu_1$ – решение "двойственной" задачи для $\{N_1, \tau_1\}, \pi_1 = (\pi', f)$; $N_2 = N', \tau_2 = \tau'$, но у требования f изменяем $r_f = \min\limits_{j \in N' \setminus f} \{r_j(\tau_2) + p_j\}, \nu_2$ – решение "двойственной" задачи для $\{N_2, \tau_2\}, \pi_2 = \pi'$.
Если после выполнения шага алгоритма получим $\{\pi_1\} = N$, т.е. все требования упорядочены, то $\pi^* = argmin\{L_{\max}(\pi_1, \tau), L_{\max}(\pi^*, \tau)\}$. Из списка примеров исключаем все примеры $\{N', \tau', \nu', \pi'\}$, для которых $\nu' \geq L_{\max}(\pi^*, \tau)$, и повторяем основной шаг алгоритма.
Алгоритм заканчивает работу, когда список примеров окажется пустым.

Нетрудно заметить, что данный алгоритм может быть использован и для решения более общей задачи $1 \mid r_j \mid \varphi_{\max}$, кроме того, работу алгоритма можно остановить и "текущее" расписание π^* будет приближённым решением исследуемой задачи. При решении тестовых примеров задачи $1 \mid r_j \mid L_{\max}$ алгоритмом 2.5 список примеров не превышал $2n^2$.

Таким образом, если исходная задача $1 \mid r_j \mid \varphi_{\max}$ является NP—трудной в сильном смысле (задача $1 \mid r_j \mid L_{\max}$ — NP—трудна в сильном смысле), то "двойственная" к ней оказалась полиноминально разрешимой.

В случае, когда между требованиями заданы отношения предшествования ациклическим графом G, то для задачи $1 \mid r_j, prec \mid \varphi_{\max}$ также может

быть поставлена и решена двойственная задача. Рассуждения и обоснование аналогично выше приведенным, — оставим их проверить заинтересованному читателю... Ключевым здесь является решение задачи $1 \mid r_j, prec \mid C_{\max}$. На последнее место расписания поочередно будут ставится требования, у которых, согласно графу отношений предшествования G, нет последователей, т.е. "двойственная" к задаче $1 \mid r_j, prec \mid \varphi_{\max}$ является полиномиально разрешимой.

При рассмотрении многоприборных задач, например $m \mid r_j, prec \mid \varphi_{\max}$, для решения "двойственной" задачи ключевым является решение NP–трудной в обычном смысле задачи РАЗБИЕНИЕ, о которой мы поговорим позже...

Таким образом, если в математическом программировании прямая и двойственная задачи, как правило, имеют одинаковый сложностной статус, то в теории расписаний оказалось, что "двойственные" задачи имеют сложностной статус меньший, чем исходные прямые задачи. Этот интересный факт требует, на наш взгляд, пристального изучения, что выходит за рамки данной книги...

2.3 “Обратная” задача

Наряду с NP-трудной задачей минимизации максимального временно́го смещения $1 \mid r_j \mid L_{\max}$, представляет интерес в некотором смысле “обратная” задача отыскания расписания π, на котором достигается максимум минимального значения временно́го смещения, и нахождение величины

$$\lambda^* = \max_{\pi \in \Pi(N)} \min_{k=\overline{1,n}} L_{j_k}(C_{j_k}(\pi)). \tag{2.25}$$

Искусственные простои прибора запрещены.

Эта задача была решена только для случая одновременного поступления требований множества N на обслуживание, т.е. $r_j = 0$, $\forall j \in N$, [13] (стр. 47–48). Рассмотрим здесь общий случай задачи $1 \mid r_j \mid \max L_{\min}$.

Лемма 2.7 *Существует оптимальное расписание $\pi = (i_1, \ldots, i_n)$ решения задачи $1 \mid r_j \mid \max L_{\min}$, для которого выполняется*

$$d_{i_k} - p_{i_k} \leq d_{i_{k+1}} - p_{i_{k+1}}, k = 2, 3, \ldots, n-1, \tag{2.26}$$

и $\lambda^* = \min_{k=\overline{1,n}} L_{i_k}(C_{i_k}(\pi))$.

Доказательство. Пусть для оптимального расписания $\pi' = (j_1, \ldots, j_n)$, $\lambda^* = \min\limits_{k=\overline{1,n}} L_{i_k}(C_{i_k}(\pi'))$, не выполняется хотя бы одно из неравенств (2.26). Далее доказательство будет состоять из двух этапов, которые могут быть несколько раз повторены.

Этап 1. Если внутри расписания π' нет простоев прибора, то перейдём к этапу 2. Пусть при расписании π' есть простои прибора, рассмотрим последний из них: $C_{j_k} < r_{j_{k+1}}$ и $r_{j_m} \le C_{j_{m-1}}, m = k+2, \ldots, n$. Построим расписание $\pi'' = (j_{k+1}, j_1, \ldots, j_k, j_{k+2}, \ldots, j_n)$. Так как $C_j(\pi'') \ge C_j(\pi'), \forall j \in N$, то значение минимального временно́го смещения не уменьшится. При расписании π'' не будет простоев прибора и сохранится оптимальное значение λ^*. Переобозначим $\pi' := \pi''$ и перейдём к этапу 2.

Этап 2. Если расписание π' удовлетворяет условиям леммы, то доказательство закончено. Пусть оказалось, что при расписании π' существует пара требований j_l, j_{l+1}, что $d_{j_l} - p_{j_l} > d_{j_{l+1}} - p_{j_{l+1}}$. Требования j_l, j_{l+1} переставим местами, получим расписание $\pi'' = (j_1, \ldots, j_{l-1}, j_{l+1}, j_l, j_{l+2}, \ldots, j_n)$. Так как при расписании π' нет простоев прибора, то $r_{j_l} \le C_{j_{l-1}}(\pi')$. Возможны случаи:

1) Пусть $r_{j_{l+1}} \le C_{j_{l-1}}(\pi')$. Очевидно, в этом случае

$$C_{j_k}(\pi') = C_{j_k}(\pi''), k = 1, 2, \ldots, l-1, l+2, \ldots, n. \tag{2.27}$$

Из предположения следует, что

$$C_{j_{l-1}}(\pi') + p_{j_l} + p_{j_{l+1}} - d_{j_{l+1}} > C_{j_{l-1}}(\pi') + p_{j_l} - d_{j_l}. \tag{2.28}$$

К тому же:

$$C_{j_{l-1}}(\pi') + p_{j_{l+1}} - d_{j_{l+1}} > C_{j_{l-1}}(\pi') + p_{j_l} - d_{j_l}; \tag{2.29}$$

$$C_{j_{l-1}}(\pi') + p_{j_{l+1}} - d_{j_l} > C_{j_{l-1}}(\pi') + p_{j_l} - d_{j_l}. \tag{2.30}$$

Из (2.27)–(2.30) видно, что значение минимального временно́го смещения не уменьшилось. Переобозначим $\pi' := \pi''$ и повторим этап 2.

2) Пусть $r_{j_{l+1}} > C_{j_{l-1}}(\pi')$. Имеем

$$C_{j_k}(\pi'') = C_{j_k}(\pi'), k = 1, 2, \ldots, l-1, \tag{2.31}$$

$$C_{j_k}(\pi'') > C_{j_k}(\pi'), k = l+2, \ldots, n. \tag{2.32}$$

Из предположений следует:

$$C_{j_{l+1}}(\pi'') - d_{j_{l+1}} > C_{j_l}(\pi') - d_{j_l}; \tag{2.33}$$

$$C_{j_l}(\pi'') - d_{j_l} > C_{j_{l+1}}(\pi') - d_{j_{l+1}}. \tag{2.34}$$

Из (2.28), (2.31)–(2.34) видно, что значение минимального временно́го смещения не уменьшилось. Переобозначим $\pi' := \pi''$ и перейдём к этапу 1.

За конечное число шагов мы придём к оптимальному расписанию, удовлетворяющему условиям леммы. Лемма доказана. □

Следующий алгоритм строит n расписаний, среди которых находится и расписание, удовлетворяющее условиям леммы.

Алгоритм 2.6 Алгоритм решения "обратной" задачи $1 \mid r_j \mid \max L_{\min}$

Пронумеруем требования множества N так, чтобы $d_1 - p_1 \le d_2 - p_2 \le \ldots \le d_n - p_n$.
Для $k = 1, 2, \ldots, n$ строим расписание $\pi_k = (k, 1, \ldots, k-1, k+1, \ldots, n)$ и находим $\lambda_k = \min\limits_{j=\overline{1,n}} L_j(C_j(\pi_k))$.
Вычисляем $\lambda^* = \max\limits_{k=\overline{1,n}} \lambda_k$.

Для перенумерации требований множества N потребуется не более $O(n \log n)$ операций. Для построения расписания π_k и нахождения λ_k, $k = 1, \ldots, n$, необходимо не более $O(n)$ операций. Следовательно, нахождение λ^* потребует не более $O(n^2)$ операций.

Решение задачи максимизации минимального временно́го смещения $1 \mid r_j \mid \max L_{\min}$ является оценкой снизу оптимального значения целевой функции NP-трудной задачи минимизации максимального временно́го смещения $1 \mid r_j \mid L_{\max}$.

Теорема 2.7 *Для оптимального значения целевой функции NP-трудной задачи $1 \mid r_j \mid L_{\max}$ решение соответствующей "обратной" задачи $1 \mid r_j \mid \max L_{\min}$ выполняется $\mu^* \ge \lambda^*$.*

Доказательство. Пусть π' и π'' – расписания решения задач $1 \mid r_j \mid L_{\max}$ и $1 \mid r_j \mid \max L_{\min}$, соответственно. Существуют требования $k', k'' \in N$, для которых:

$$\mu^* = C_{k'}(\pi') - d_{k'} \ge C_j(\pi') - d_j, \quad \forall j \in N; \tag{2.35}$$

$$C_j(\pi'') - d_j \ge C_{k''}(\pi'') - d_{k''} = \lambda^*, \quad \forall j \in N. \tag{2.36}$$

Требования k' и k'' могут совпадать. Пусть расписание $\pi'' = (j_1, \ldots, j_n)$, для требования j_1, очевидно, имеет место

$$C_{j_1}(\pi') - d_{j_1} \ge C_{j_1}(\pi'') - d_{j_1}. \tag{2.37}$$

Из (2.35), (2.37), (2.36) будем иметь

$$\mu^* = C_{k'}(\pi') - d_{k'} \ge C_{j_1}(\pi') - d_{j_1} \ge C_{j_1}(\pi'') - d_{j_1} \ge C_{k''}(\pi'') - d_{k''} = \lambda^*,$$

т.е. $\mu^* \geq \lambda^*$. Теорема доказана. $\square$

Таким образом, для решения NP-трудной задачи $1 \mid r_j \mid \varphi_{\max}$ при неубывающих функциях штрафа предлагается алгоритм 2.5, реализующий метод ветвей и границ. Оценка снизу решения подпримеров осуществляется с помощью решения "двойственной" задачи, которое может быть найдено за число операций не превышающее $O(n^2)$.

Кроме "двойственной" задачи решена ещё "обратная" задача, алгоритм 2.6 трудоёмкости $O(n^2)$ операций. Если в задаче минимизации максимального временно́го смещения "стараются выравнять" смещения минимизируя максимальное, то в "обратной" задаче смещения "выравниваются" за счёт максимизации минимального. Естественно, искусственные простои прибора запрещены.

На наш взгляд, необходимо использовать весь богатый арсенал математического программирования для решения задач теории расписаний...

Глава 3

Алгоритмы оптимального решения задачи $1 \mid r_j \mid L_{\max}$

В данной главе мы предложим два точных алгоритма решения общего случая задачи $1 \mid r_j \mid L_{\max}$. Также будет рассмотрен алгоритм для решения одного варианта задачи распознавания задачи $1 \mid r_j \mid L_{\max}$.

Разработка точных алгоритмов для общего случая рассматриваемой задачи началась в 70-х годах 20-го века. Из работ того времени нам известны алгоритмы Десокого и Маргентайллера (Dessouky, Margenthaler [113]), Братлей и др. (Bratley et al. [92]), Бейкера и Шу (Baker, Su [80]), Макмахона и Флориана (McMahon, Florian [173]). Последние два алгоритма были также рассмотрены в работе Лейджвега и др. (Lageweg et al. [142]). Было экспериментально показано, что данные алгоритмы являются наиболее эффективными среди упомянутых четырех. Следующим этапом развития точных методов решения задачи $1 \mid r_j \mid L_{\max}$ является работа Карлье (Carlier [101]). Предложенный им алгоритм на основе оригинальной схемы ветвления до сих пор считается лучшим алгоритмом для точного решения рассматриваемой задачи. В литературе можно найти несколько работ, где алгоритм Карлье применяется в качестве подалгоритма при решении более сложных задач, например в [70, 85].

Разработанные алгоритмы решения задачи $1 \mid r_j \mid L_{\max}$ во многом основаны на идеях, заложенных в алгоритме Карлье, а также в методе программирования в ограничениях (*Constraint Programming*). Для полноты изложения данные алгоритм и метод описаны в разделе 3.1. Нужно отметить, что будет показан вариант алгоритма Карлье, несколько изменённый для улучшения эффективности. Разработанные два алгоритма решения рассматриваемой задачи представлены в разделе 3.2.

В разделе 3.3 продемонстрируем результаты экспериментального сравнения четырёх алгоритмов решения общего случая задачи $1 \mid r_j \mid L_{\max}$, двух

представленных в данной работе алгоритмов, а также алгоритмов Карлье, Макмахона и Флориана. Насколько нам известно, сравнение последних двух алгоритмов никогда не проводилось.

В разделе 3.4 будет рассмотрен вариант задачи распознавания для задачи $1 \mid r_j \mid L_{\max}$. В данном варианте задаётся пример задачи и значение максимального временно́го смещения. Необходимо узнать, существует ли допустимое расписание, т.е. расписание с максимальным временны́м смещением, которое не больше заданного значения. В случае отрицательного ответа, требуется найти как можно меньшее подмножество требований, для которого также не существует допустимого расписания. Алгоритм, представленный для решения данного варианта задачи, будет использован в главе 4.

В данной главе предполагается, что все параметры требований $\{r_j, p_j, d_j\}$, $j \in N$, являются целыми числами.

3.1 Существующие методы решения задачи $1 \mid r_j \mid L_{\max}$

3.1.1 Алгоритм Карлье

В данном подразделе рассмотрим алгоритм Карлье [101], который на данный момент считается одним из наиболее эффективных точных методов решения задачи $1 \mid r_j \mid L_{\max}$. В целях улучшения алгоритм был нами несколько изменён. Будем продолжать называть изменённый алгоритм алгоритмом Карлье, так как произведённые изменения не столь принципиальны.

Следует заметить, что оригинальная статья [101] содержала некоторые неточности, в следствие которых предложенный алгоритм был некорректен. Однако, в [111, 178] была произведена необходимая модификация оригинального алгоритма, обеспечивающая его корректность.

В основе алгоритма Карлье лежит алгоритм Шраге (Schrage [192]) сложности $O(n \log n)$, который выполняется на каждом узле дерева поиска. В алгоритме Шраге на каждое последующее место в расписание ставится требование с минимальным директивным сроком среди уже поступивших на обслуживание (если таких нет, то выбирается требование среди имеющих минимальное время поступления среди неупорядоченных). Иногда этот алгоритм называют модифицированным алгоритмом Джексона. Опишем алгоритм формально (алгоритм 3.1). В алгоритме предполагается, что требования пронумерованы по неубыванию времён поступления, что не ограничивает общности. Оптимальным расписанием π_{Sch} является раннее расписание, соответствующее найденной перестановке τ.

Алгоритм 3.1 Алгоритм Шраге (Schrage)

1: $t := r_1$; $N' := N$; $\tau := \emptyset$
2: **while** $N' \neq \emptyset$ **do**
3: $\quad V := \{j \mid j \in N', r_j \leq t\}$
4: $\quad$ **if** $V = \emptyset$ **then**
5: $\quad\quad V := \min\{j \mid j \in N'\}$
6: $\quad$ **end if**
7: $\quad i := \arg\min_{j \in V}\{d_j\}$
8: $\quad \tau := (\tau, i)$; $N' := N' \setminus \{i\}$; $t := \max\{t, r_i\} + p_i$
9: **end while**

Следующая лемма определяет нижнюю оценку значения максимального временно́го смещения оптимального расписания.

Лемма 3.1 *[101] Для любого множества* $N' \subseteq N$

$$h(N') = \min_{j \in N'} r_j + \sum_{j \in N'} p_j - \max_{j \in N'} d_j = r_{N'} + p_{N'} - d_{N'}$$

является нижней оценкой оптимального значения максимального временно́го смещения. □

Лемма 3.1 используется в алгоритме для нахождения нижних оценок, а также в теореме 3.1, которая обосновывает критерий оптимальности расписания Шраге и правило ветвления алгоритма Карлье.

Теорема 3.1 *Пусть построено расписание Шраге* $\pi_{Sch} \in \Pi(N)$ *для данного примера задачи, и требования пронумерованы в том порядке, в котором они упорядочены в* π_{Sch}. *Пусть также* $b \in N$ *– критическое требование, т.е.* $L_b(\pi_{Sch}) = L_{\max}(\pi_{Sch})$, *и* $a \in N$ *– последнее такое требование, что* $a \leq b$ *и*

$$s_a(\pi_{Sch}) - \min_{a \leq j \leq b} r_j \leq L_b(\pi_{Sch}) - UB, \tag{3.1}$$

где UB *– некоторая верхняя оценка значения оптимального решения задачи,* $UB \leq L_{\max}(\pi_{Sch})$. *Примем* $S = \{a, \ldots, b\}$, *тогда:*

- *если не существует такого требования* $c \in S$, *что* $d_c > d_b$, *то для данного примера не существует расписания с максимальным временны́м смещением, меньшим, чем* UB;
- *иначе, если* $c \in S$ — *наибольший такой индекс, что* $d_c > d_b$, *то в любом расписании* π', *для которого* $L_{\max}(\pi') < UB$, *требование* c *выполняется или до, или после всех требований из множества* $J = \{c + 1, \ldots, b\}$.

Доказательство. Заметим, что между требованиями из множества S в расписании π_{Sch} прибор не простаивает. Если бы прибор простаивал, это означало бы, что существует требование $a' \in S$, $a' > a$, обслуживаемое сразу после простоя и удовлетворяющее ограничению (3.1), так как $s_{a'}(\pi_{Sch}) = r_{a'} = \min\limits_{j \in S} r_j$ (согласно правилу построения расписания Шраге). И это бы противоречило условию, что требование a последнее такое требование.

Поэтому,

$$L_p(\pi_{Sch}) = s_a(\pi_{Sch}) + \sum_{j \in S} p_j - d_b. \tag{3.2}$$

Если не существует такого требования $c \in S$, что $d_c > d_b$, то $d_b = \max\limits_{j \in S} d_j$. Используя (3.1) и (3.2), имеем:

$$\min_{j \in S} r_j + \sum_{j \in S} p_j - \max_{j \in S} d_j \geq UB. \tag{3.3}$$

Согласно лемме 3.1, левая часть неравенства (3.3) является нижней оценкой оптимального решения для данного примера, т.е. не существует такого расписания $\pi' \in \Pi(N)$, что $L_{\max}(\pi) < UB$.

Пусть теперь существует требование $i \in S$, $d_i > d_b$. И пусть c – последнее такое требование, поэтому $d_b = \max\limits_{j \in J} d_j$. Аналогично предыдущему случаю, так как между требованиями из S в π_{Sch} отсутствуют простои, мы имеем $L_b(\pi_{Sch}) = s_c(\pi_{Sch}) + p_c + \sum\limits_{j \in J} p_j - d_b$. Заметим, что $s_c(\pi_{Sch}) < \min\limits_{j \in J} r_j$, иначе существовало бы требование из J, обслуживаемое перед требованием c в расписании Шраге π_{Sch}, так как $d_c > d_b = \max\limits_{j \in J} d_j$. Пусть требование c выполняется между требованиями из J в π'. Тогда временно́е смещение требования $k \in J \subset N$, обслуживаемого последним среди требований из $J \cup \{c\}$ составляет $L_k(\pi') \geq \min\limits_{j \in J} r_j + p_c + \sum\limits_{j \in J} p_j - d_k \geq L_p(\pi_{Sch}) \geq UB$, так как $d_k \leq \max\limits_{j \in J} d_j < d_p$. Поэтому в расписании π' требование c обслуживается или до, или после всех требований из J. □

Теорема 3.1 отличается от аналогичной теоремы из [101] способом определения требования a (в [101] требование a – ближайшее слева к b в π_{Sch} требование, перед которым прибор простаивает). Предлагаемый нами альтернативный способ позволяет несколько сократить перебор.

Правило ветвления алгоритма Карлье основано на теореме 3.1. После построения расписания Шраге, если выполняется второй случай теоремы 3.1, мы находим требование c и множество требований J. Далее создаются два дочерних узла, в одном из которых требование c обслуживается перед

всеми требованиями из J (для этого изменяется его директивный срок на $d_c = d_b - p_J$), а в другом требование c обслуживается после требований из множества J (для этого изменяется его время поступления на $r_c = r_J + p_J$). Если выполняется первый случай теоремы 3.1, то для текущего примера не существует лучшего расписания, чем уже найденное с максимальным временны́м смещением UB. Поэтому ветвления из этого узла не происходит.

Правило оценивания алгоритма основано на лемме 3.1. Каждому узлу дерева поиска соответствует нижняя оценка LB. После построения расписания Шраге нижняя оценка дочерних узлов определяется как максимум от LB, $h(J)$ и $h(J \cup \{c\})$. Для оценивания снизу может использоваться решение соответствующей "двойственной" и/или обратной задачи. Дочерний узел рассматривается только, если его нижняя оценка меньше чем текущая верхняя оценка UB.

В качестве *стратегии поиска* используется поиск "в глубину", причём из дочерних узлов первым рассматривается тот, где требование c обслуживается после всех требований из J.

Проведённое нами предварительное экспериментальное исследование показало, что добавление нижней оценки, используемой в алгоритме Макмахона и Флориана, увеличивает эффективность алгоритма. Заметим, что вычисление нижней оценки Макмахона и Флориана также основано на построении расписания Шраге.

Формально алгоритм Карлье представлен как алгоритм 3.2.

3.1.2 Метод программирования в ограничениях

В данном подразделе мы рассмотрим метод программирования в ограничениях (*ПвО*) и его использование для решения задач теории расписаний.

Программирование в ограничениях (*Constraint Programming*) – это метод, созданный для решения комбинаторных задач, которые могут быть смоделированы как одна или несколько *задач выполнимости системы ограничений* (*ЗВСО*) (*Constraint Satisfaction Problem*). Пример ЗВСО задаётся множеством *переменных*, множеством возможных значений для каждой переменной, называемым *доменом* (*domain*), а также множеством *ограничений*, заданных на множестве переменных. ЗВСО является задачей распознавания, где требуется узнать, существуют ли такие возможные значения переменных, при которых выполняются все заданные ограничения. Множество таких значений переменных, если задача выполнима, называется *решением* ЗВСО.

Алгоритм 3.2 Алгоритм Карлье

$\pi^* := \emptyset$; $UB := \infty$
CARLIER(N,$-\infty$)

procedure CARLIER(N,LB)
построим расписание Шраге $\pi_{Sch} \in \Pi(N)$
if $L_{\max}(\pi_{Sch}) < UB$ **then**
  $\pi^* := \pi_{Sch}$; $UB := L_{\max}(\pi_{Sch})$
end if
перенумеруем требования в том порядке, в котором они обслуживаются в π_{Sch}
найдём критическое требование $b \in N$: $L_b(\pi_{Sch}) = L_{\max}(\pi_{Sch})$
$LB := \max\{LB, LB_{MF}, h(J) = r_J + p_J - d_b\}$
найдём наибольший такой номер $a \in N$, что $a \le b$ и
$s_a(\pi_{Sch}) - \min\limits_{a \le j \le b} r_j \le L_b(\pi_{Sch}) - UB$
if $LB < UB$ **AND** существует такое требование $i \in \{a, \ldots, b\}$, что $d_i > d_b$ **then**
  выберем наибольший такой номер $c \in \{a, \ldots, b\}$, что $d_c > d_b$
  $J := \{c+1, \ldots, b\}$
  $LB_1 := \max\{LB, r_J + p_J + p_c - d_c\}$
  if $LB_1 < UB$ **then**
    $\hat{r} := r_c$; $r_c := r_J + p_J$; **CARLIER**(N,LB_1); $r_c = \hat{r}$
  end if
  $LB_2 := \max\{LB, r_c + p_J + p_c - d_b\}$
  if $LB_2 < UB$ **then**
    $\hat{d} := d_c$; $d_c := d_b - p_J$; **CARLIER**(N,LB_2); $d_c := \hat{d}$
  end if
end if
end procedure

В ПвО ограничения используются не только для тестирования допустимости решения, но также и в активном режиме для удаления значений из доменов переменных (сокращение доменов), определения вытекающих новых ограничений, установления несовместимости. Процесс использования ограничений для получения этих заключений называется *пропагацией ограничений* (*constraint propagation*). Алгоритмы, которые осуществляют пропагацию ограничений, называются *алгоритмами фильтрации* (*filtering algorithms*).

Так как в общем случае ЗВСО является NP-трудной задачей [7], обычно пропагация ограничений не даёт ответа на вопрос о существовании решения задачи. То есть не все следствия из системы ограничений могут быть выведены. В частности, одной пропагацией ограничений не всегда можно установить несовместимость системы. Поэтому для решения задачи необходимо дополнительно производить частичный перебор решений. Обычно это осуществляется с помощью алгоритма, использующего дерево поиска. В итоге метод ПвО является в некоторой степени аналогом метода ветвей и границ. Отличие заключается в том, что в методе ветвей и границ для сокращения перебора используется правило оценивания, а в ПвО – пропагация ограничений.

Важный принцип ПвО, так называемый "принцип локальности", заключается в том, что пропагация каждого ограничения должна осуществляться как можно более независимо от других ограничений задачи. В соответствии с данным принципом алгоритмы фильтрации разрабатываются для каждого ограничения или группы ограничений отдельно, что позволяет не создавать специальные алгоритмы для каждой вновь возникающей задачи, а использовать уже существующие алгоритмы как "кирпичики" для решения сложных практических задач.

Детальное представление метода программирования в ограничениях можно найти в книгах [75, 171].

Задачи теории расписаний для одного прибора с запрещением прерываний могут быть смоделированы как ЗВСО следующим образом. Для каждого требования $j \in N$ вводится переменная $start(j)$, которая обозначает время фактического начала обслуживания требования j. Также для каждого требования в качестве исходных данных задачи задаются: время поступления требования r_j, жесткий директивный срок $\bar{d}_j$ (*deadline*), а также время обслуживания p_j. Напомним, что все исходные данные задачи должны быть целыми числами. Любое требование $j \in N$ может обслуживаться только в отрезке $[r_j, \bar{d}_j]$. Поэтому начальным доменом переменной $start(j)$ является множество целых чисел из отрезка $[r_j, \bar{d}_j - p_j]$ (Рисунок 3.1). В процессе реше-

ния задачи домен переменной $start(j)$ может уменьшаться, но для удобства мы будем использовать r_j и $\bar{d}_j$ также для обозначения текущего наименьшего времени начала обслуживания (*текущего времени поступления*) и текущего наибольшего времени окончания обслуживания (*текущего предельного срока*) требования j, соответственно.

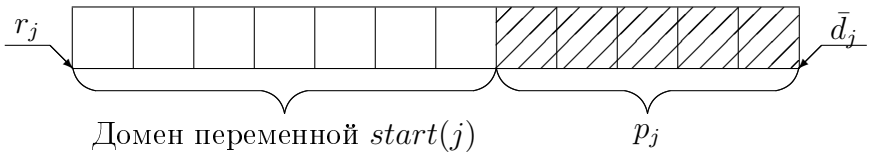

Рис. 3.1: Требование j и домен переменной $start(j)$

"Глобальное" ограничение[1] `disjunctive` определяется на множестве переменных $start(j)$, $j \in N$, и обеспечивает выполнение не более одного требования в каждый момент времени, т.е. для каждой пары требований $i, j \in N$ гарантируется выполнение ограничения

$$(start(i) + p_i \leq start(j)) \vee (start(j) + p_j \leq start(i)).$$

Мы рассмотрим наиболее используемую технику фильтрации для "глобального" ограничения `disjunctive`, названную в англоязычной литературе *"Edge-Finding"*. Данная техника предназначена для установления неявных отношений предшествования между некоторым требованием $k \in N$ и некоторым множеством требований $\Omega \subset N$, т.е. устанавливается требование k, которое должно обслуживаться после всех требований из множества Ω ($\Omega \to k$) или перед всеми требованиями из Ω ($k \to \Omega$). Данное заключение может вести к изменению минимального времени начала обслуживания или максимального времени окончания обслуживания требования k, т.е. к уменьшению домена переменной $start(k)$. Напомним, что для произвольного множества требований S введены обозначения $r_S = \min\limits_{j \in S} r_j$, $d_S = \max\limits_{j \in S} d_j$, $p_S = \sum\limits_{j \in S} p_j$. Следующие правила раскрывают суть техники "Edge-Finding":

$$\forall \Omega, \forall k \notin \Omega : [\bar{d}_\Omega - r_{\Omega \cup \{k\}} < p_\Omega + p_k] \Rightarrow [\Omega \to k]; \tag{3.4}$$

$$\forall \Omega, \forall k \notin \Omega : [\Omega \to k] \Rightarrow [start(k) \geq \max_{\emptyset \neq \Omega' \subseteq \Omega} \{r_{\Omega'} + p_{\Omega'}\}]; \tag{3.5}$$

$$\forall \Omega, \forall k \notin \Omega : [\bar{d}_{\Omega \cup \{k\}} - r_\Omega < p_\Omega + p_k] \Rightarrow [k \to \Omega]; \tag{3.6}$$

$$\forall \Omega, \forall k \notin \Omega : [k \to \Omega] \Rightarrow [start(k) \leq \min_{\emptyset \neq \Omega' \subseteq \Omega} \{\bar{d}_{\Omega'} - p_{\Omega'}\} - p_k]. \tag{3.7}$$

При наличии n требований всего априори существует $O(n2^n)$ пар (k, Ω) для проверки условий (3.4) и (3.6). Мы рассмотрим алгоритм, детально представленный как алгоритм 3.3, осуществляющий все возможные изменения

[1] В ПвО ограничение называется "глобальным", если оно состоит из нескольких простых ограничений.

доменов переменных $start(j)$, $j \in N$, согласно правилам (3.4)–(3.5) за время $O(n^2)$. Данный алгоритм описан в книге Баптиста и др. (Baptiste et al. [84, с. 24]). Можно заметить, что, заменив все времена поступления r_j на $r'_j = d_N - d_j$, а все предельные сроки $\bar{d}_j$ на $\bar{d}'_j = d_N - r_j$, из условий (3.4) и (3.5) мы получим, соответственно, (3.6) и (3.7). Поэтому алгоритм может быть также применён для изменения доменов переменных $start(j)$, $j \in N$, согласно правилам (3.6)–(3.7), так как примеры $\{r_j, p_j, \bar{d}_j\}$ и $\{r'_j, p_j, \bar{d}'_j\}$, $j \in N$ эквивалентны согласно лемме 1.3.

Алгоритм 3.3 Алгоритм фильтрации "Edge-Finding" для "глобального" ограничения `disjunctive`

for $l := 1$ to n **do**
  $r'_l := r_l$
end for
for $l := 1$ to n **do**
  $P := 0$; $C := -\infty$; $H := -\infty$
  for $k := n$ down to 1 **do**
    if $\bar{d}_k \leq \bar{d}_l$ **then**
      $P := P + p_k$; $C := \max\{C, r_k + P\}$
    end if
    $C_k := C$
  end for
  for $k := 1$ to n **do**
    if $\bar{d}_k \leq \bar{d}_l$ **then**
      $H := \max\{H, r_k + P\}$; $P := P - p_k$
    else
      if $r_k + p_k + P > \bar{d}_l$ **then**
        $r'_k := \max\{r'_k, C_k\}$
      end if
      if $H + p_k > \bar{d}_l$ **then**
        $r'_k := \max\{r'_k, C\}$
      end if
    end if
  end for
end for
for $l := 1$ to n **do**
  $r_l := r'_l$
end for

В качестве данных алгоритм получает текущие времена поступления r_j, текущие предельные сроки $\bar{d}_j$ и продолжительности обслуживания p_j, т.е. домены $[r_j, \bar{d}_j - p_j]$ переменных $start(j)$, $j \in N$. На выходе домены могут быть сокращены путём увеличения текущих времён поступления. Для алгоритма подразумевается, что множество требований $N = \{1, 2, \ldots, n\}$ отсортировано по неубыванию r_j. Через r'_j обозначаются изменённые в процессе работы алгоритма значения r_j.

Внешний цикл алгоритма работает по текущим предельным срокам требований $\bar{d}_j$. На l-той итерации рассматривается l-тое требования в порядке неубывания r_j. Затем множество требований разбивается на подмножество $\Omega(l)$, содержащее требования с текущим предельным сроком не более $\bar{d}_l$, и подмножество $\bar{\Omega}(l) = N \setminus \Omega(l)$. Изменения текущих времён поступления будут происходить у требований, входящих в последнее подмножество. Пусть также $\Omega(l,k) = \{j \in \Omega(l) \mid r_j \geq r_k\}$.

После первого внутреннего цикла, т.е. после строки 11 алгоритма 3.3, мы располагаем следующими значениями:

$$\begin{aligned} P &= p_{\Omega(l)}; \\ C &= \max_{\emptyset \neq \Omega' \subseteq \Omega(l)} \{r_{\Omega'} + p_{\Omega'}\}; \\ C_k &= \max_{\emptyset \neq \Omega' \subseteq \Omega(l,k)} \{r_{\Omega'} + p_{\Omega'}\}, \forall k \in N. \end{aligned}$$

На строках 16–18 алгоритма 3.3 происходят изменения текущих времён поступления согласно правилам (3.4) и (3.5) для требования k и всех возможных подмножеств множества $\Omega(l,k)$. Благодаря строке 19 переменная алгоритма P равняется $p_{\Omega(l,k)}$. Если значение $r_k + p_k + P$ превышает $\bar{d}_l$, то $\Omega(l,k) \to k$, и поэтому r_k может быть увеличено до $\max\{r'_k, C_k\}$.

На строках 19–21 алгоритма 3.3 происходят изменения текущих времён поступления согласно правилам (3.4) и (3.5) для требования k и всех возможных подмножеств множества Ω, включающих по меньшей мере одно требование j, идущее перед требованием k в порядке неубывания текущих времён поступления, т.е. $j < k$. Благодаря условию в строке 19 переменная алгоритма H равняется максимальному из минимальных времён окончания обслуживания всех требований из таких подмножеств, т.е. $H = \max\limits_{j<k} r_j + p_{\Omega(l,j)}$. Пусть $j_{\max} = \arg\max\limits_{j<k} r_j + p_{\Omega(l,j)}$. Если значение $H + p_k$ превышает $\bar{d}_l$, то $\Omega(l, j_{\max}) \to k$ и r_k может быть увеличено до $\max\{r'_k, C_{j_{\max}}\}$. По определению $C_{j_{\max}} = C$.

Другие варианты алгоритмов фильтрации, действующие согласно технике “Edge-Finding”, были рассмотрены в работах Карлье и Пинсона (Carlier, Pinson [102, 103]). Алгоритм, представленный в последней из этих работ, имеет трудоёмкость $O(n \log n)$ операций, однако требует использование гораздо более сложных структур данных.

3.2 Алгоритмы решения задачи $1 \mid r_j \mid L_{\max}$

В данном разделе мы представим два алгоритма решения общего случая задачи $1 \mid r_j \mid L_{\max}$. Первый алгоритм построен по методу программирования в ограничениях, в котором используется специальное правило ветвления. Второй алгоритм основан на алгоритме Карлье. Существенным отличием предлагаемого алгоритма является использование алгоритма фильтрации "Edge-Finding" на каждом узле дерева поиска. За счёт включения данного алгоритма достигается значительное сокращение перебора и увеличение эффективности алгоритма.

Задача $1 \mid r_j \mid L_{\max}$ может быть смоделирована как ЗВСО при помощи ограничения `disjunctive`, рассмотренного в предыдущем подразделе. В самом деле, проверка выполнимости данного "глобального" ограничения является вариантом распознавания рассматриваемой задачи. При фиксированном значении максимального временно́го смещения L' вариант распознавания заключается в проверке существования расписания $\pi \in \Pi(N)$, максимальное временно́е смещение которого не превышает заданное значение: $L_{\max}(\pi) \leq L'$. Данная задача формулируется как ЗВСО с одним "глобальным" ограничением `disjunctive` с исходными данными $\{r_j, p_j, \bar{d}_j\}$, $\forall j \in N$, где жесткий директивный срок каждого требования $j \in N$ задаётся как $\bar{d}_j = d_j + L'$.

Как уже было сказано выше, для построения алгоритма, решающего ЗВСО по методу ПвО, необходимо определить три составные части: как будет осуществляться пропагация ограничений, правило ветвления и стратегию поиска. В нашем случае пропагация единственного ограничения осуществляется алгоритмом 3.3.

Правило ветвления может задаваться несколькими стандартными способами. Во-первых, ветвление может происходить по домену какой-либо переменной, когда в каждом из дочерних узлов в домене выбранной переменной остается по одному значению. То есть создаётся такое число дочерних узлов текущего узла дерева поиска, сколько значений было в текущем домене выбранной переменной. В данном случае стратегия выбора переменной для ветвления также оказывает заметное влияние на эффективность всего алгоритма. Во-вторых, ветвление может происходить по отношениям предшествования. В данном случае выбирается множество требований $\Omega \subseteq N$ и в дочерних узлах добавляется ограничение, согласно которому одно из требований из Ω обслуживается первым (последним) среди требований из Ω. То есть создаётся $\mid \Omega \mid$ дочерних узлов. Заметим, что в данном случае в потомках текущего узла дерева поиска должна осуществляться пропагация не только

начального ограничения `disjunctive`, но и ограничений предшествования.

В нашем случае, однако, может быть применена более эффективная специализированная схема ветвления, заимствованная из алгоритма Карлье. Когда алгоритмы фильтрации ограничения `disjunctive` не могут далее сократить домены переменных, выполняется алгоритм Шраге. Если в полученном расписании Шраге время окончания обслуживания не превышает "жёсткий" директивный срок для всех требований, то допустимое решение ЗВСО найдено. Если нижняя оценка, основанная на построенном расписании, строго больше нуля, то легко убедиться в том, что допустимого решения ЗВСО не существует. Если же существование допустимого решения ЗВСО остаётся под вопросом, то ветвление происходит таким же образом, как и в алгоритме Карлье, согласно теореме 3.1. Как и в алгоритме 3.2, в дочерних узлах дерева поиска изменяется или время поступления, или жёсткий директивный срок определённого требования c. В терминах ПвО такое изменение соответствует сокращению домена переменной $start(c)$.

В ПвО стратегия "лучшей оценки" не может быть применена в качестве стратегии поиска, так как в ЗВСО отсутствует целевая функция и, соответственно, правило оценивания. Поэтому в алгоритмах по методу ПвО обычно используется поиск "в глубину". При этом стратегия должна определять порядок, в котором просматриваются дочерние узлы.

Последним шагом в построении алгоритма решения задачи $1 \mid r_j \mid L_{\max}$ по методу ПвО является моделирование целевой функции. Задача минимизации целевой функции может быть решена путём последовательного решения нескольких ЗВСО, алгоритм для решения которых уже определён. Перед этим необходимо определить верхнюю и нижнюю оценки целевой функции ub и lb. Это может быть сделано, например, алгоритмом Шраге. Верхняя оценка определяется максимальным временны́м смещением построенного расписания Шраге. Нижняя оценка может быть получена аналонично тому, как это происходит в алгоритме Макмахона и Флориана. Может быть использовано в качестве нижней оценки решение соответствующей "двойственной" и/или обратной задачи. После определения оценок целевой функции существуют следующие способы нахождения оптимума.

- Решаются примеры ЗВСО, последовательно увеличивая значение L' на 1 начиная с нижней оценки lb, пока для некоторого значения L^* ЗВСО не окажется разрешимой. Данное значение L^* и будет оптимальным значением, а решением задачи будет решение последней ЗВСО.
- Решаются примеры ЗВСО, последовательно уменьшая значение L' на 1 начиная с верхней оценки ub, пока для некоторого значения L^* ЗВСО не

окажется неразрешимой. Значение L^*+1 будет оптимальным значением, а решением задачи будет решение последней разрешимой ЗВСО.

- Дихотомия. Осуществляется повторением двух этапов:

  - решается ЗВСО с $L' = \lfloor (ub + lb)/2 \rfloor$;
  - если ЗВСО разрешима, то $ub := L'$, если неразрешима, то $lb := L'+1$.

  Данный процесс продолжается до тех пор, пока верхняя оценка ub не сравняется с нижней lb. Значение $L^* = ub = lb$ и будет оптимальным значением целевой функции. Решением задачи будет решение примера ЗВСО с $L' = L^*$.

Проведённые нами эксперименты показали, что при решении задачи $1 \mid r_j \mid L_{\max}$ алгоритмом по методу ПвО использование дихотомии даёт наименьшее среднее число решённых ЗВСО. Соответственно, при использовании дихотомии среднее время решения меньше, чем при использовании первых двух способов.

Для вычисления начальных значений верхней и нижней оценок ub и lb выполняется алгоритм Шраге. Максимальное временно́е смещение расписания Шраге является начальной верхней оценкой. Начальной нижней оценкой является нижняя оценка Макмахона и Флориана LB_{MF} [173].

Формально описанный метод представлен как алгоритм 3.4, где используется функция решения ЗВСО (алгоритм 3.5).

Алгоритм 3.4 Алгоритм решения задачи $1 \mid r_j \mid L_{\max}$ по методу ПвО

```
1: π* := ∅;
2: построим расписание Шраге и найдем начальные оценки ub и lb = LB_MF
3: while lb < ub do
4:    sol := ⌊(ub + lb)/2⌋
5:    d̄_j := d_j + sol, ∀j ∈ N
6:    if CSP(r,p,d̄) then
7:       ub := sol
8:    else
9:       lb := sol + 1
10:   end if
11: end while
```

Недостатком приведённого алгоритма, основанного на методе ПвО, является необходимость решения нескольких ЗВСО. В общем случае сложность решения ЗВСО соответствует сложности решения всего примера исходной задачи. Более того, примеры ЗВСО не сильно отличаются друг от друга, что приводит к многократному выполнению одних и тех же действий.

Алгоритм 3.5 Функция решения ЗВСО для алгоритма по методу ПвО

function CSP($\bar{r}$,$\bar{p}$,$\bar{d}$)
$d^* := \max_{j \in N} \bar{d}_j$
repeat
  положим $r_j := \bar{r}_j$, $p_j := \bar{p}_j$, $d_j := \bar{d}_j$, $\forall j \in N$, и выполним алгоритм 3.3
  $\bar{r} := r_j$, $\forall j \in N$
  положим $r_j := d^* - \bar{d}_j$, $p_j := \bar{p}_j$, $d_j := d^* - \bar{r}_j$, $\forall j \in N$, и выполним алгоритм 3.3
  $\bar{d}_j := d^* - r_j$, $\forall j \in N$
until нет изменений в $\bar{r}$ или $\bar{d}$
построим расписание Шраге $\pi_{Sch} \in \Pi(N)$ для примера $\{\bar{r}, \bar{p}, \bar{d}\}$,
if $L_{\max}(\pi_{Sch}) \leq 0$ **then**
  $\pi^* := \pi_{Sch}$; **return** $True$
end if
перенумеруем требования в том порядке, в котором они обслуживаются в π_{Sch}
найдём критическое требование $b \in N$: $L_b(\pi_{Sch}) = L_{\max}(\pi_{Sch})$
$LB := \max\{LB_{MF}, h(J) = \bar{r}_J + \bar{p}_J - \bar{d}_b\}$;
найдём наибольший такой номер $a \in N$, что $a \leq b$ и
$s_a(\pi_{Sch}) - \min_{a \leq j \leq b} \bar{r}_j < L_b(\pi_{Sch})$
$rs := False$
if $LB \leq 0$ **AND** существует такое требование $i \in \{a, \ldots, b\}$, что $\bar{d}_i > \bar{d}_b$ **then**
  выберем наибольший такой номер $c \in \{a, \ldots, b\}$, что $\bar{d}_c > \bar{d}_b$
  $J := \{c+1, \ldots, b\}$
  if $\max\{LB, \bar{r}_J + \bar{p}_J + \bar{p}_c - \bar{d}_c\} \leq 0$ **then**
    $\hat{r} := \bar{r}_c$; $\bar{r}_c := \bar{r}_J + \bar{p}_J$; $rs := rs$ **or CSP**($\bar{r}$,$\bar{p}$,$\bar{d}$); $\bar{r}_c := \hat{r}$
  end if
  if $\max\{LB, \bar{r}_c + \bar{p}_J + \bar{p}_c - \bar{d}_b\} \leq 0$ **then**
    $\hat{d} := \bar{d}_c$; $\bar{d}_c := \bar{d}_b - \bar{p}_J$; $rs := rs$ **or CSP**($\bar{r}$,$\bar{p}$,$\bar{d}$); $\bar{d}_c := \hat{d}$
  end if
end if
return rs
end function

Для решения задачи $1 \mid r_j \mid L_{\max}$ существует возможность отказаться от использования стандартной схемы метода ПвО и использовать алгоритмы фильтрации без решения последовательности ЗВСО. Для этого применяется алгоритм ветвей и границ, в котором ветвление происходит за счёт увеличения времён поступлений и/или уменьшения директивных сроков. На каждом узле дерева поиска перед ветвлением алгоритмы фильтрации выполняются для дальнейшего изменения r_j и d_j. Для этого директивные сроки требований временно изменяются на величину, меньшую на 1, чем текущая верхняя оценка UB: $\bar{d}_j := d_j + UB - 1$. Затем алгоритмы фильтрации сокращают насколько это возможно домены $[r_j, \bar{d}_j]$ переменных $start(j)$, $j \in N$. Если при этом определена несовместимость, то текущий узел дерева поиска обрезается, так как лучшего решения, чем UB не существует, как это делается и в алгоритмах ветвей и границ. Если несовместимость не доказана, то директивные сроки получают обновленные значения $d_j := \bar{d}'_j - UB + 1$, $j \in N$, и ветвление происходит обычным способом (времена поступления обновляются алгоритмами фильтрации напрямую).

Предварительные эксперименты показали, что наиболее эффективным выбором является использование алгоритмов фильтрации на узлах дерева поиска алгоритма Карлье. Опять же, целесообразно применять и технику фильтрования “Edge-Finding”, так как дополнительное использование фильтрации “Not-First”/“Not-Last” не даёт сокращения перебора.

Формально описанный алгоритм представлен как алгоритм 3.6.

3.3 Экспериментальное сравнение алгоритмов решения задачи $1 \mid r_j \mid L_{\max}$

В данном разделе мы представим результаты экспериментального исследования следующих четырёх алгоритмов:

- **MMF** – Алгоритм Макмахона и Флориана;
- **CAR** – Алгоритм 3.2 (Карлье с использованием нижней оценки Макмахона и Флориана);
- **CP** – Алгоритм 3.4 (по методу ПвО);
- **HYB** – Алгоритм 3.6 (метод ветвей и границ с использованием правила ветвления Карлье, нижней оценки Макмахона и Флориана и алгоритма фильтрации “Edge-Finding”).

Алгоритм 3.6 Алгоритм решения задачи $1 \mid r_j \mid L_{\max}$ с использованием правила ветвления Карлье и фильтрации "Edge-Finding"

$\pi^* := \emptyset$; построим расписание Шраге и найдём начальные оценки UB и LB
NODE(LB,r,p,d)

procedure NODE(LB,$\bar{r}$,$\bar{p}$,$\bar{d}$)
$d^* := \max_{j \in N} \bar{d}_j + UB$
repeat
  положим $r_j := \bar{r}_j$, $p_j := \bar{p}_j$, $d_j := \bar{d}_j + UB - 1$, $\forall j \in N$, и выполним алгоритм 3.3
  $\bar{r} := r_j$, $\forall j \in N$
  положим $r_j := d^* - \bar{d}_j$, $p_j := \bar{p}_j$, $d_j := d^* - \bar{r}_j + UB - 1$, $\forall j \in N$, и выполним алгоритм 3.3
  $\bar{d}_j := d^* - r_j$, $\forall j \in N$
until нет изменений в $\bar{r}$ или $\bar{d}$
построим расписание Шраге $\pi_{Sch} \in \Pi(N)$ для примера $\{\bar{r}, \bar{p}, \bar{d}\}$,
if $L_{\max}(\pi_{Sch}) < UB$ **then**
  $\pi^* := \pi_{Sch}$; $UB := L_{\max}(\pi_{Sch})$
end if
перенумеруем требования в том порядке, в котором они обслуживаются в π_{Sch}
найдём критическое требование $b \in N$: $L_b(\pi_{Sch}) = L_{\max}(\pi_{Sch})$
$LB := \max\{LB_{MF}, h(J) = \bar{r}_J + \bar{p}_J - \bar{d}_b\}$;
найдём наибольший такой номер $a \in N$, что $a \leq b$ и $s_a(\pi_{Sch}) - \min\limits_{a \leq j \leq b} \bar{r}_j < L_b(\pi_{Sch})$
if $LB < UB$ **AND** существует такое требование $i \in \{a, \ldots, b\}$, что $\bar{d}_i > \bar{d}_b$ **then**
  выберем наибольший такой номер $c \in \{a, \ldots, b\}$, что $\bar{d}_c > \bar{d}_b$
  $J := \{c+1, \ldots, b\}$
  $LB_1 := \max\{LB, \bar{r}_J + \bar{p}_J + \bar{p}_c - \bar{d}_c\}$
  if $LB_1 < UB$ **then**
    $\hat{r} := \bar{r}_c$; $\bar{r}_c := \bar{r}_J + \bar{p}_J$; **NODE**($LB_1$,$\bar{r}$,$\bar{p}$,$\bar{d}$); $\bar{r}_c := \hat{r}$
  end if
  $LB_2 := \max\{LB, \bar{r}_c + \bar{p}_J + \bar{p}_c - \bar{d}_b\}$
  if $LB_2 < UB$ **then**
    $\hat{d} := \bar{d}_c$; $\bar{d}_c := \bar{d}_b - \bar{p}_J$; **NODE**($LB_2$,$\bar{r}$,$\bar{p}$,$\bar{d}$); $\bar{r}_c := \hat{r}$
  end if
end if
end procedure

Как было показана в лемме 1.3, инверсные примеры I^P и I^Q с параметрами требований, соответственно, $\{r_j^P, p_j, d_j^P\}$ и $\{r_j^Q = -d_j^P, p_j, d_j^Q = -r_j^P\}$, $j \in N$, являются эквивалентными. Однако, сложность решения двух инверсных примеров может сильно отличаться.

Обозначим $R = \max_{j \in N} r_j - \min_{j \in N} r_j, \qquad D = \max_{j \in N} d_j - \min_{j \in N} d_j$. В [142] было экспериментально показано, что алгоритм Макмахона и Флорина эффективнее решает примеры, для которых выполняется $R \geq D$. Наши предварительные тесты также показали, что все четыре исследуемых алгоритма работают быстрее для примеров, для которых выполняется $R \geq D$. Поэтому основной эксперимент проводился с алгоритмами, соответственно, **MMF_RD**, **CAR_RD**, **CP_RD**, **HYB_RD**, которые решают инверсный эквивалентный пример, если в заданном примере $R < D$.

Для экспериментов были использованы три набора тестовых примеров. Примеры из первых двух наборов "l" и "s" были сгенерированы следующим образом (здесь K_r и K_d — параметры генерации).

- Продолжительности обслуживания p_j генерировались по равномерному распределению в отрезке $[10, 100]$.
- Времена поступления r_j генерировались по равномерному распределению в отрезке $[1, K_r n]$.
- Для примеров из набора "l" директивные сроки d_j генерировались по равномерному распределению в отрезке $[1, K_d n]$.
- Для примеров из набора "s" директивные сроки d_j генерировались по равномерному распределению в отрезке $[r_j + p_j, r_j + p_j + K_d n]$.

Все параметры требований являются целыми числами.

Для обоих наборов тестовых примеров были использованы следующие значения параметров.

n	50, 100, 150, 200, 250, 300
K_r	20, 30, 40, 50, 60
K_d	20, 30, 40, 50, 60

Для каждой тройки (n, K_r, K_d) и каждого из двух наборов было сгенерировано 100 примеров, т.е. и набор "l", и набор "s" содержат по 2500 примеров для каждой размерности. Предварительные эксперименты показали, что данный выбор значений параметров K_r и K_d задаёт наиболее "трудную" область примеров. Данное наблюдение подтверждает результаты, представленные в [101, 142].

Третий набор тестовых примеров “h” включает в себя 85 “трудных” примеров, встретившихся при решении задачи $1 \mid r_j \mid \sum w_j U_j$ методом ветвей и отсечений (раздел 4.5). Данные примеры содержат от 44 до 116 требований.

Эксперименты проводились на компьютере с процессором 2 GHz и памятью 512 Mb. Введём следующие обозначения полученных результатов:

$P_{1\mathrm{m}}$ – процент примеров, оптимально решённых за 1 минуту;

T_{av} – среднее время в миллисекундах, необходимое для оптимального решения примера;

N_{av} – среднее число просмотренных узлов в дереве поиска.

Заметим, что при достижении лимита времени (1 минута) для подсчёта результатов используется текущее время и текущее число просмотренных узлов в дереве поиска. То есть при наличии нерешённых примеров приведённые значения T_{av} и N_{av} являются нижними оценками реальных значений.

В таблице 3.1 представлены результаты для тестовых наборов примеров “l” и “s”. Алгоритм **MMF_RD** оказался худшим по всем параметрам. С его помощью удалось решить около 97.5% примеров. Алгоритм **CAR_RD** показал себя намного эффективнее, он не смог решить только 0.22% тестовых примеров. Напомним, что примеры генерировались из “трудной” области, поэтому эффективность алгоритма можно признать удовлетворительной для решения примеров на практике. Однако заметим, что из 66 примеров размерности от 50 до 300, для которых оптимальное решение не было найдено за 1 минуту, только для 8 примеров алгоритм **CAR_RD** нашёл оптимальное решение за время, меньшее чем 30 минут.

Каждый из алгоритмов **CP_RD** и **HYB_RD** решил все тестовые примеры за 1 минуту. Причём последнему потребовалось больше 4 секунд для решения только одного из 30000 примеров. Алгоритму **HYB_RD**, как и ожидалось, в среднем, требуется просмотреть меньше узлов, чем алгоритму **CP_RD**, так как последнему требуется решить несколько ЗВСО. В экспериментах данный алгоритм рассматривал в среднем 3 ЗВСО для решения одного примера. При сравнимом времени, затрачиваемом на решение одного узла дерева поиска алгоритм **HYB_RD** превосходит по скорости решения алгоритм **CP_RD**. Заметим также, что оптимизация реализации этих двух алгоритмов может существенно сократить время их исполнения.

О полном превосходстве алгоритмов, использующих пропагацию, говорить, однако, преждевременно. В данных алгоритмах время, затрачиваемое на рассмотрение одного узла дерева поиска, намного превосходит таковое для

Таблица 3.1: Сравнение результатов для тестовых наборов “l” и “s”

	MMF_RD			**CAR_RD**		
Test	P_{1m}	T_{av}	N_{av}	P_{1m}	T_{av}	N_{av}
l-50	94.24%	3523.2	151842.5	99.84%	109.4	7192.1
l-100	96.32%	2213.3	31048.3	99.60%	247.9	5431.2
l-150	97.48%	1546.2	10668.5	99.68%	194.0	1971.1
l-200	97.88%	1278.7	5438.1	99.76%	148.3	956.6
l-250	97.56%	1474.5	4229.2	99.60%	248.5	1061.7
l-300	97.84%	1314.5	2853.4	99.64%	228.6	596.1
s-50	98.12%	1225.6	53015.0	100%	4.2	209.1
s-100	97.20%	1761.9	25659.7	99.96%	28.8	559.4
s-150	98.00%	1217.2	8908.1	99.68%	209.5	1860.4
s-200	98.40%	1002.4	4302.6	99.84%	108.0	497.9
s-250	98.52%	896.4	2716.9	99.88%	81.5	282.0
s-300	99.04%	581.9	1304.0	99.88%	86.8	213.4
	CP_RD			**HYB_RD**		
Test	P_{1m}	T_{av}	N_{av}	P_{1m}	T_{av}	N_{av}
l-50	100%	11.7	18.6	100%	3.4	5.6
l-100	100%	64.6	32.0	100%	18.2	10.0
l-150	100%	187.4	43.8	100%	55.2	14.5
l-200	100%	407.0	56.2	100%	119.0	18.7
l-250	100%	732.0	66.7	100%	225.6	23.3
l-300	100%	1179.6	75.6	100%	373.2	27.4
s-50	100%	5.6	8.5	100%	1.7	2.9
s-100	100%	29.3	13.9	100%	8.6	4.6
s-150	100%	90.8	19.8	100%	27.9	6.9
s-200	100%	148.1	20.1	100%	40.3	6.2
s-250	100%	230.6	20.6	100%	63.7	6.5
s-300	100%	380.7	23.8	100%	105.2	7.5

алгоритмов **MMF_RD** и **CAR_RD**. Поэтому, хотя и среднее число узлов в алгоритмах **CP_RD** и **HYB_RD** намного меньше, среднее время решения для примеров большой размерности сравнимо с таковым для алгоритма Карлье.

В целом, можно сделать заключение о том, что, если требуется обязательное точное решение всего множества заданных примеров задачи $1 \mid r_j \mid L_{\max}$ размерности до нескольких сотен требований, например для решения подзадач в некотором алгоритме, вариант **HYB_RD** является предпочтительным. Если же не требуется точного решения всего множества примеров, то можно посоветовать использование алгоритма **CAR_RD**.

Таблица 3.2: Сравнение результатов для тестового набора “h”

CAR_RD			**HYB_RD**		
$P_{1\text{m}}$	T_{av}	N_{av}	$P_{1\text{m}}$	T_{av}	N_{av}
96.47%	1588.6	28879.2	100%	620.1	386.6

В заключении мы приведём небольшую иллюстрацию того, что с “трудными” примерами задачи $1 \mid r_j \mid L_{\max}$ действительно можно столкнуться на практике. В таблице 3.2 приведены результаты решения набора “трудных” примеров, встретившихся при решении задачи $1 \mid r_j \mid \sum w_j U_j$ (набор тестовых примеров “h”). Как видно из таблицы, все примеры достаточно быстро решены алгоритмом **HYB_RD**, но не алгоритмом **CAR_RD**.

3.4 Модифицированный алгоритм Карлье

В данном разделе мы рассмотрим вариант задачи распознавания для задачи минимизации максимального временно́го смещения для одного прибора.

Назовем расписание π *допустимым* по отношению к константе L', если $L_{\max}(\pi) \leq L'$. Множество требований S *допустимо* по отношению к константе L', если существует допустимое расписание $\pi \in \Pi(S)$, и *недопустимо*, если не существует такого расписания.

Пусть нам дано множество требований N с временами поступления, продолжительностями обслуживания, директивными сроками для каждого требования, а также значение L'. В рассматриваемом здесь варианте задачи необходимо определить, допустимо или нет множество N по отношению к заданной константе. Более того, если множество N недопустимо, то требуется найти недопустимое подмножество требований $S \subseteq N$, наименьшее по количеству требований $\mid S \mid$.

Для решения поставленной задачи предлагается алгоритм метода ветвей и границ, который мы будем для краткости называть “модифицированный алгоритм Карлье”, так как он является производным от алгоритма Карлье, представленного в подразделе 3.1.1. Данный алгоритм является эвристическим в том смысле, что он находит некоторое недопустимое подмножество требований S, не обязательно минимальное, когда множество требований N недопустимо. В тоже время, алгоритм точно определяет, допустимо ли множество требований N.

В предлагаемом алгоритме на каждом узле дерева поиска мы строим расписание Шраге согласно алгоритму 3.1. Если на каком-то узле построенное расписание оказалось допустимым, то задача выполнена и алгоритм прекращает свою работу. Следующая теорема 3.2 определяет свойства недопустимого расписания Шраге, согласно которым в алгоритме происходит отсечение текущего узла дерева поиска или ветвление. На рисунках 3.2 и 3.3 проиллюстрированы, соответственно, случаи 1 и 2, рассмотренные в теореме.

Теорема 3.2 *Пусть дано недопустимое по отношению к константе L' расписание Шраге $\pi_{Sch} \in \Pi(N)$ и требования пронумерованы в том порядке, в котором они упорядочены в π_{Sch}. Пусть также $b \in N$ – такое требование, что $L_b(\pi_{Sch}) > L'$, и $a \in N$, – последнее такое требование что $a \leq b$ и*

$$\alpha = s_a(\pi_{Sch}) - \min_{j \in S} r_j < L_b(\pi_{Sch}) - L' = \beta, \tag{3.8}$$

где $S = \{a, \ldots, b\}$. Тогда:

1) *если не существует такого требования $c \in S$, что $d_c > d_b$, то множество требований S недопустимо по отношению к L';*

2) *иначе, если $c \in S$ – последнее требование с директивным сроком $d_c > d_b$, и существует допустимое расписание $\pi' \in \Pi(N)$, то в данном расписании π' требование c выполняется или до, или после всех требований из множества $J = \{c+1, \ldots, b\}$.*

Доказательство. 1) Во-первых, заметим, что между требованиями из множества S в расписании π_{Sch} прибор не простаивает. Если бы прибор простаивал, это означало бы, что существует требование $a' \in S$, обслуживаемое после a и удовлетворяющее ограничению (3.8), так как $s_{a'}(\pi_{Sch}) = r_{a'} = \min\limits_{j \in S} r_j$ (согласно правилу построения расписания Шраге π_{Sch}). И это бы противоречило условию, что требование a последнее такое требование. Поэтому,

$$L_p(\pi_{Sch}) = s_a(\pi_{Sch}) + \sum_{j \in S} p_j - d_b. \tag{3.9}$$

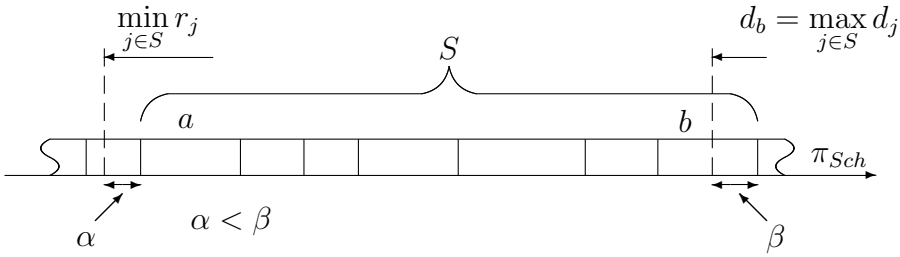

Рис. 3.2: Расписание Шраге: недопустимое подмножество S.

Расписание обслуживания требований множества S является примером незадерживающей цепочки, которые мы рассматривали ранее. Так как не существует такого требования $c \in S$, что $d_c > d_b$, то $d_b = \max\limits_{j\in S} d_j$. Используя (3.8) из (3.9) получим

$$\min_{j\in S} r_j + \sum_{j\in S} p_j - \max_{j\in S} d_j > L'. \tag{3.10}$$

Согласно лемме 3.1, левая часть неравенства (3.10) является нижней оценкой максимального временно́го смещения любого расписания из $\Pi(S)$, т.е. не существует такого расписания $\pi' \in \Pi(S)$, что $\max\limits_{j\in S} L_j(\pi) \le L'$. Следовательно, множество требований S недопустимо. Заметим, что требование всегда можно найти, так как первое требование в π_{Sch} всегда удовлетворяет условию (3.8).

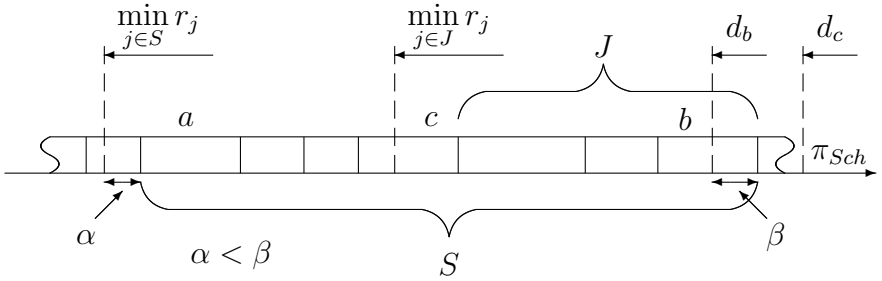

Рис. 3.3: Расписание Шраге: требование c и множество требований J.

2) Так как c – последнее такое требование из S, что $d_c > d_b$, то $d_b = \max\limits_{j\in J} d_j$. Аналогично предыдущему случаю, так как между требованиями из S в π_{Sch} отсутствуют простои, имеем $L_b(\pi_{Sch}) = s_c(\pi_{Sch}) + p_c + \sum\limits_{j\in J} p_j - d_b$. Заметим, что $s_c(\pi_{Sch}) < \min\limits_{j\in J} r_j$, иначе существовало бы требование из J, обслуживаемое раньше требования c в расписании Шраге π_{Sch} (так как $d_c > d_b = \max\limits_{j\in J} d_j$). Пусть расписание $\pi' \in \Pi(N)$ допустимо и требование c выполняется между требованиями из J в π'. Тогда временно́е смещение требования $k \in J \subset N$, обслуживаемого последним среди требований из $J \cup \{c\}$ составля-

ет $L_k(\pi') \geq \min\limits_{j\in J} r_j + p_c + \sum\limits_{j\in J} p_j - d_k \geq L_p(\pi_{Sch}) > L'$, так как $d_k \leq \max\limits_{j\in J} d_j < d_p$, и это противоречит условию допустимости расписания $\max\limits_{j\in N} L_j(\pi') \leq L'$. Поэтому в расписании π' требование c обслуживается или до, или после всех требований из J. □

Правило ветвления алгоритма основано на теореме 3.2. Если построенное расписание Шраге недопустимо, мы находим требование c и множество требований J. Далее ветвление происходит также, как это осуществляется в алгоритме Карлье. *Правило оценивания* и *стратегия поиска* обоих алгоритмов также аналогичны.

Отличительной чертой модифицированного алгоритма Карлье является способ построения недопустимого подмножества требований. На каждом узле дерева поиска, где не происходит ветвления, согласно теореме 3.2, мы располагаем подмножеством S, недопустимым для задачи с текущими параметрами требований. Данное подмножество передаётся "родительскому" узлу дерева поиска. Способ построения недопустимого подмножества для узла, имеющего дочерние узлы, обоснован в теореме 3.3. В конце работы алгоритма вершина дерева поиска вернёт недопустимое подмножество требований для исходного примера. Процесс работы алгоритма схематично изображён на рисунке 3.4.

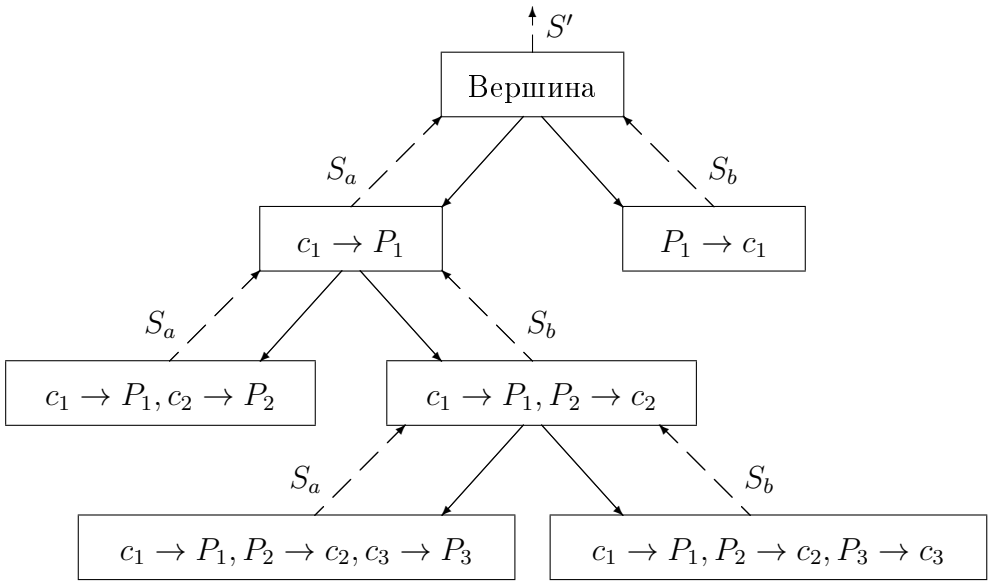

Рис. 3.4: Дерево поиска модифицированного алгоритма Карлье

Теорема 3.3 *Пусть в текущей задаче в любом допустимом расписании* $\pi' \in \Pi(N)$ *требование* $c \in N$ *может быть обслужено только до или после*

всех требований из множества $J \subset N$. Пусть также $S_a \subset N$ – недопустимое множество требований для задачи с дополнительным ограничением, согласно которому требование c обслуживается перед всеми требованиями из множества J, а $S_b \subset N$ – недопустимое подмножество для задачи, где c выполняется после всех требований из J. Тогда:

1) если $c \notin S_a$, то множество требований S_a недопустимо для текущей задачи;

2) если $c \notin S_b$, то множество требований S_b недопустимо для текущей задачи;

3) если $c \in S_a$ и $c \in S_b$, то множество требований $S' = J \cup S_a \cup S_b$ недопустимо для текущей задачи.

Доказательство. 1) Если $c \notin S_a$, то на требования из множества S_a не наложено никаких дополнительных ограничений в задаче, где требование c обслуживается перед всеми требованиями из J. Поэтому множество требований S_a будет также недопустимым и для текущей задачи.

2) Доказательство аналогично случаю 1.

3) Докажем от противного. Допустим, что существует допустимое расписание $\pi \in \Pi(S')$. Так как $c \in S'$ и $J \subset S'$, то имеются три возможности постановки требования c в расписание π. Во-первых, c может обслуживаться перед всеми требованиями из J, но в этом случае подмножество требований $S_a \subset S$ недопустимо. Во-вторых, c может выполняться после всех требований из J в π, но в этом случае подмножество требований $S_b \subset S$ недопустимо. Следовательно, c должно быть обслужено между требованиями из J в расписании π, но это противоречит условию теоремы. Поэтому, не существует допустимого расписания $\pi \in \Pi(S')$ и множество требований S' недопустимо. □

Формально модифицированный алгоритм Карлье представлен как алгоритм 3.7. Алгоритм получает в качестве исходных данных множество требований N и константу L' и возвращает или допустимое расписание π^*, или недопустимое подмножество требований S'.

Пример. Рассмотрим работу модифицированного алгоритма Карлье на примере, состоящем из 9 требований. Параметры требований приведены в Таблице 3.3. Константа L' равна 0. *Узел 1.* Вершина. $S = \{1, \ldots, 3\}$, $c = 2$. Первый

дочерний узел 2: положим $r_2 := 4$, получаем $S_a = \{4, \ldots, 8\}$. Второй дочерний узел 5: положим $d_2 := 4$, получаем $S_b = \{1, 2\}$. Так как $c = 2 \notin S_a$, то данный узел возвращает множество требований $S' := S_a = \{4, \ldots, 8\}$.

Алгоритм 3.7 Модифицированный алгоритм Карлье

$\pi^* := \emptyset$
MOD_CARLIER(N, S')

procedure MOD_CARLIER(N, S')
построим расписание Шраге $\pi_{Sch} \in \Pi(N)$
if $L_{\max}(\pi_{Sch}) < L'$ **then**
  $\pi^* := \pi_{Sch}$; алгоритм заканчивает работу
end if
перенумеруем требования в том порядке, в котором они обслуживаются в π_{Sch}
найдём наименьшее такое требование $b \in N$, что $L_b(\pi_{Sch}) > L'$
найдём наибольший такой номер $a \in N$, стоящее перед b в π_{Sch},
что $s_a(\pi_{Sch}) - \min\limits_{j \in S} r_j < L_b(\pi_{Sch}) - L'$, где $S = \{a, \ldots, b\}$
if существует такое требование $i \in \{a, \ldots, b\}$, что $d_i > d_b$ **then**
  выберем наибольший такой номер $c \in \{a, \ldots, b\}$, что $d_c > d_b$
  $J := \{c + 1, \ldots, b\}$
  $\hat{r} := r_c$; $r_c := r_J + p_J$; **MOD_CARLIER**(N, S_b); $r_c := \hat{r}$
  $\hat{d} := d_c$; $d_c := d_c - p_J$; **MOD_CARLIER**(N, S_a); $d_c := \hat{d}$
  if $c \notin S_a$ **then**
    $S' := S_a$
  else if $c \notin S_a$ **then**
    $S' := S_b$
  else
    $S' := J \cup S_a \cup S_b$
  end if
else
  $S' := \{a, \ldots, b\}$; **exit**
end if
end procedure

Таблица 3.3: Параметры требований

j	1	2	3	4	5	6	7	8	9
r_j	0	1	3	6	7	11	12	19	16
p_j	2	3	1	3	3	2	5	3	4
d_j	4	8	5	11	22	17	18	23	30

Узел 2. Родительский узел: 1. $S = \{4, \ldots, 7\}$, $c = 5$. Первый дочерний узел 3: положим $r_5 := 18$, получаем $S_a = \{5, 8\}$. Второй дочерний узел 4: положим $d_5 := 11$, получаем $S_b = \{4, 5\}$. Так как $c = 5 \in S_a$ и $c = 5 \in S_b$, данный узел возвращает множество требований $S' := J \cup S_a \cup S_b = \{4, \ldots, 8\}$.

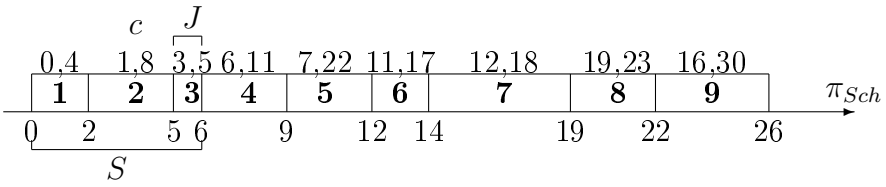

Рис. 3.5: Узел 1.

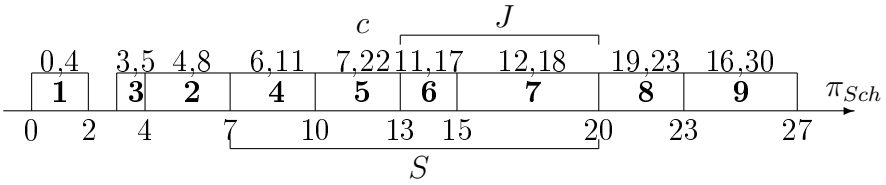

Рис. 3.6: Узел 2.

Узел 3. Родительский узел: 2. $S = \{5, 8\}$, нет c. Данный узел возвращает множество требований $S' := S = \{5, 8\}$.

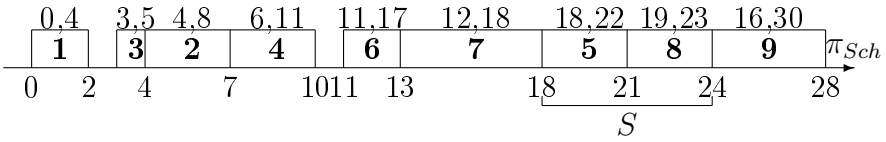

Рис. 3.7: Узел 3.

Узел 4. Родительский узел: 2. $S = \{4, 5\}$, нет c. Данный узел возвращает множество требований $S' := S = \{4, 5\}$.

Узел 5. Родительский узел: 1. $S = \{1, 2\}$, нет c. Данный узел возвращает множество требований $S' := S = \{1, 2\}$.

Так как из вершины дерева поиска мы получили множество требований $S' = \{4, \ldots, 8\}$, то S' недопустимо. В данном случае множество S' также является *минимальным* недопустимым подмножеством, т.е. при удалении любого требования, множество S' становится допустимым. Дерево поиска, построенного алгоритмом при решении рассматриваемого примера с 9 требованиями, изображено на рисунке 3.10.

Модифицированный алгоритм Карлье будет нами использоваться в главе 4 для построения отсечений в алгоритме ветвей и проверок.

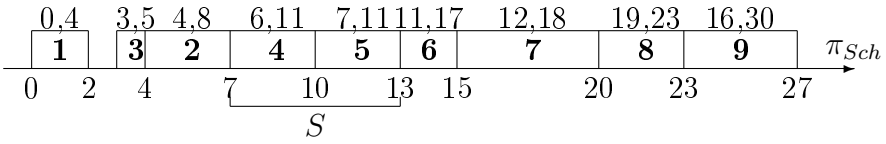

Рис. 3.8: Узел 4.

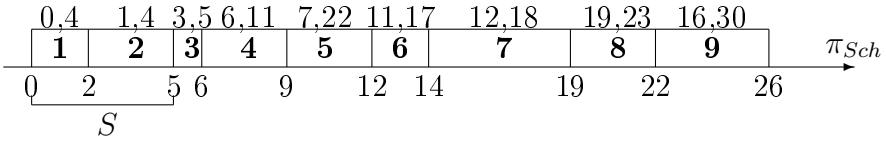

Рис. 3.9: Узел 5.

3.5 Эффективные алгоритмы решения задачи минимизации максимального временно́го смещения

Предлагаются полиномиальные и псевдополиномиальные алгоритмы решения частных случаев задачи минимизации максимального временно́го смещения, а также приближённые алгоритмы с оценкой абсолютной погрешности оптимального значения целевой функции. Для общей задачи $1|r_j|L_{\max}$ разработан новый псевдополиномиальный приближённый алгоритм, использующий алгоритм решения одного из частных случаев задачи, и получена оценка абсолютной погрешности оптимального значения целевой функции.

3.5.1 Процедура построения приближённого решения задачи $1|r_i \leq r_j, d_i \geq d_j|L_{\max}$. Оценка абсолютной погрешности

В этом параграфе предлагается процедура построения приближённого расписания для задачи $1|r_j|L_{\max}$ при ограничениях на параметры требований (1.63) ($r_i \leq r_j \;\Rightarrow\; d_i \geq d_j \;\;\forall \;\; i,j \in N$) с величиной границы абсолютной

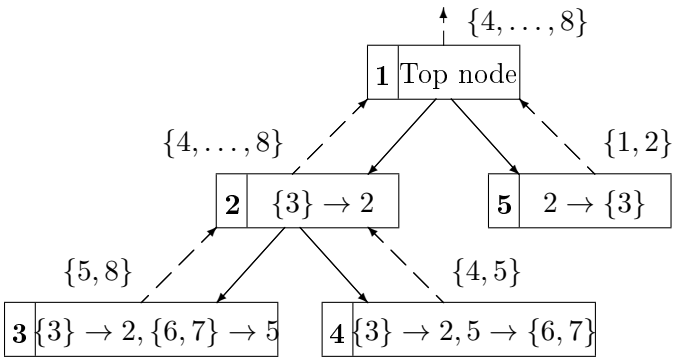

Рис. 3.10: Пример дерева поиска, построенного алгоритмом 3.7

погрешности оптимального значения целевой функции, не превышающей величины максимальной продолжительности обслуживания требований.

Опишем ниже процедуру h_0 построения расписания $\pi_{h_0} \in \Pi(N,t)$. Значение целевой функции расписания π_{h_0} будет отличаться от оптимального значения не более, чем на величину максимальной продолжительности обслуживания требований. Кроме того, момент завершения обслуживания всех требований расписания π_{h_0} будет отличаться от момента завершения обслуживания всех требований расписания, оптимального по быстродействию, также не более, чем на $p_{\max}(N)$.

Положим $P = \max\{r_{\max}(N), t\} + \sum\limits_{j=1}^{n} p_j - \max\{r_{\min}(N), t\}$.

Процедура состоит из n итераций. Внутри итерации для каждого целочисленного момента времени, число которых не превосходит P, следующее приближённое расписание получается добавлением требования из множества неупорядоченных с максимальным моментом поступления и минимальным директивным сроком к уже построенному расписанию.

Первоначально перенумеруем требования множества N таким образом, чтобы соблюдались неравенства

$$r_1 \geq r_2 \geq ... \geq r_n, \quad d_1 \leq d_2 \leq ... \leq d_n. \tag{3.11}$$

Алгоритм 3.8 Процедура h_0.

Полагаем $\bar{t} = \max\{r_n, t\}$, $N_1 = \{1\}$, $P_1 = \max\{r_1, t\} + \sum\limits_{j \in N} p_j - p_1$,
$\pi_1^i = (1)$, $\pi_1^i \in \Pi(N_1, i)\ \forall\ i = \overline{\bar{t}, P_1}$.
Пусть уже известны N_k, P_k, π_k^i для всех $i = \overline{\bar{t}, P_k}$, и $1 \leq k < n$.
Полагаем $N_{k+1} = N_k \cup \{k+1\}$, $P_{k+1} = P_k - p_{k+1}$,
$\pi_i' = (k+1, \pi_k^{\max\{r_{k+1}, i\}+p_{k+1}})$,
$\pi_i'' = (\pi_k^i, k+1)$,
$\pi_i''' = (k+1, \pi_k^i)$,
$\pi_{k+1}^i = \arg\min\{C_{\max}(\pi) | \pi \in \Pi_i\}$,
$\Pi_i = \{\pi \in \{\pi_i', \pi_i'', \pi_i'''\} : L_{\max}(\pi) = \min\limits_{\bar{\pi} \in \{\pi_i', \pi_i'', \pi_i'''\}} L_{\max}(\bar{\pi})\}$,
$\pi_i', \pi_i'', \pi_i''', \pi_{k+1}^i \in \Pi(N_{k+1}, i)\ \forall\ i = \overline{\bar{t}, P_{k+1}}$.
При $k = n$ полагаем $\pi_{h_0} = \pi_k^{\bar{t}}$, и процесс заканчивается.

Очевидно, что в процедуре h_0 (алгоритм 3.8) перенумерация требований множества N согласно (3.11) возможна в силу условий (1.63) ($r_i \leq r_j \Rightarrow d_i \geq d_j\ \forall\ \ i, j \in N$).

Поскольку $r_{k+1} \leq r_1$ и $\bar{t} \leq i \leq P_k$, то $\max\{r_{k+1}, i\} + p_{k+1} \leq P_k$. Кроме того, $P_{k+1} \leq P_k$. Поэтому расписания $\pi_k^{\max\{r_{k+1}, i\}+p_{k+1}}$ и π_k^i, $i = \overline{\bar{t}, P_{k+1}}$, которые

используются для построения π^i_{k+1}, $k = 1, ..., n-1$, будут уже построены на предыдущей итерации процедуры.

Сформулируем свойства расписаний, построенных процедурой h_0 в случае, если на параметры требований множества N накладываются ограничения (1.63) ($r_i \le r_j \;\Rightarrow\; d_i \ge d_j \;\;\forall\;\; i, j \in N$). Покажем, что момент завершения обслуживания всех требований расписания $\pi_{h_0} \in \Pi(N, t)$, построенного процедурой h_0, не превосходит момента завершения обслуживания всех требований оптимального по быстродействию расписания на величину $p_{\max}(N)$.

Лемма 3.2 *Пусть расписание $\pi_{h_0} \in \Pi(N, t)$ построено процедурой h_0, и выполняются условия* (1.63) ($r_i \le r_j \;\Rightarrow\; d_i \ge d_j \;\;\forall\;\; i, j \in N$). *Тогда справедливо неравенство $C_{\max}(\pi_{h_0}) - C_{\max}(\vec{\pi}_r) \le p_{\max}(N)$ для любого $\vec{\pi}_r \in \vec{\Pi}_r(N, t)$.*

Доказательство. Покажем, что при всех $m = 1, ..., n$ выполняется

$$C_{\max}(\pi^i_m) - C_{\max}(\vec{\pi}^i_m) \le p_{\max}(N) \;\;\forall\;\; i = \bar{t}, ..., P_m, \tag{3.12}$$

где расписание $\pi^i_m \in \Pi(N_m, i)$ построено в процедуре h_0, $\vec{\pi}^i_m \in \vec{\Pi}_r(N_m, i)$. Пусть $m = 1$. Очевидно, что $C_{\max}(\pi^i_1) = C_{\max}(\vec{\pi}^i_1)$, где $\pi^i_1 = (1) \in \Pi(N_1, i)$, $\vec{\pi}^i_1 \in \vec{\Pi}_r(N_1, i)\; \forall\; i = \bar{t}, ..., P_1$, а значит, неравенства (3.12) выполняются. Пусть неравенства (3.12) выполняются при $m = k$, где $1 \le k \le n-1$. Докажем, что (3.12) справедливы при $m = k+1$. Тогда неравенства (3.12) будут доказаны для всех $m = 1, ..., n$. Покажем, что

$$C_{\max}(\pi'_i) - C_{\max}(\vec{\pi}^i_{k+1}) \le p_{\max}(N) \;\;\forall\;\; i = \bar{t}, ..., P_{k+1} \tag{3.13}$$

и

$$C_{\max}(\pi'''_i) - C_{\max}(\vec{\pi}^i_{k+1}) \le p_{\max}(N) \;\;\forall\;\; i = \bar{t}, ..., P_{k+1}, \tag{3.14}$$

где $\pi'_i = (k+1, \pi_k^{\max\{r_{k+1}, i\}+p_{k+1}}) \in \Pi(N_{k+1}, i)$, $\pi'''_i = (k+1, \pi^i_k) \in \Pi(N_{k+1}, i)$.

Так как расписание π'_i строится с момента времени i ($i = \bar{t}, \ldots, P_m$) добавлением к требованию $(k+1)$ расписания $\pi_k^{\max\{r_{k+1}, i\}+p_{k+1}} \in \Pi(N_k, i)$ с момента его завершения, равного $\max\{r_{k+1}, i\} + p_{k+1}$, то

$$C_{\max}(\pi'_i) = C_{\max}(\pi_k^{\max\{r_{k+1}, i\}+p_{k+1}}) \;\;\forall\;\; i = \bar{t}, ..., P_{k+1}. \tag{3.15}$$

Согласно (3.11) $r_{k+1} \le r_j \;\forall\; j \in N_k$. Кроме того, поскольку $\vec{\pi}^i_{k+1} \in \Pi_r(N_{k+1}, i)$ и $\vec{\pi}_k^{\max\{r_{k+1}, i\}+p_{k+1}} \in \vec{\Pi}_r(N_k, \max\{r_{k+1}, i\} + p_{k+1})$, то выполняется $(k+1, \vec{\pi}_k^{\max\{r_{k+1}, i\}+p_{k+1}}) \in \vec{\Pi}_r(N_{k+1}, i)$. Значит,

$$C_{\max}(\vec{\pi}^i_{k+1}) = C_{\max}(\vec{\pi}_k^{\max\{r_{k+1}, i\}+p_{k+1}}) \;\;\forall\;\; i = \bar{t}, ..., P_{k+1}. \tag{3.16}$$

С учётом (3.15), (3.16) выполняется

$$C_{\max}(\pi'_i) - C_{\max}(\vec{\pi}^i_{k+1}) = C_{\max}(\pi_k^{\max\{r_{k+1},i\}+p_{k+1}}) - C_{\max}(\vec{\pi}_k^{\max\{r_{k+1},i\}+p_{k+1}}).$$

. Отсюда и из (3.12) при $m = k$ следует неравенство (3.13) для $m = 1, \ldots, n$.

Докажем теперь справедливость (3.14). Пусть $\pi^i_k = (j_1, ..., j_{n_k})$, $n_k = |N_k|$. Отметим, что порядок обслуживания требований множества N_k одинаков при расписаниях $\pi_i^{'''}$ и π^i_k для всех $i = \bar{t}, ..., P_{k+1}$, а моменты начала их обслуживания равны $\max\{\max\{r_{k+1}, i\}+p_{k+1}, r_{j_1}\}$ и $\max\{r_{j_1}, i\}$, соответственно. Поскольку $\max\{\max\{r_{k+1}, i\} + p_{k+1}, r_{j_1}\} \geq \max\{r_{j_1}, i\}\ \forall\ i = \bar{t}, ..., P_{k+1}$, то

$$C_{\max}(\pi_i^{'''}) \geq C_{\max}(\pi^i_k). \tag{3.17}$$

Пусть в (3.17) для некоторого $\bar{t} \leq i \leq P_{k+1}$ $C_{\max}(\pi_i^{'''}) = C_{\max}(\pi^i_k)$. Тогда с учётом (3.12) при $m = k$ и (3.16) $C_{\max}(\pi_i^{'''}) - C_{\max}(\vec{\pi}^i_{k+1}) = C_{\max}(\pi^i_k) - C_{\max}(\vec{\pi}_k^{\max\{r_{k+1},i\}+p_{k+1}}) \leq C_{\max}(\vec{\pi}^i_k) + p_{\max}(N) - C_{\max}(\vec{\pi}_k^{\max\{r_{k+1},i\}+p_{k+1}})$. Так как $\vec{\pi}^i_k \in \vec{\Pi}_r(N_k, i)$, $\vec{\pi}_k^{\max\{r_{k+1},i\}+p_{k+1}} \in \vec{\Pi}_r(N_k, \max\{r_{k+1}, i\} + p_{k+1})$ и $i \leq \max\{r_{k+1}, i\} + p_{k+1}$, то $C_{\max}(\vec{\pi}^i_k) \leq C_{\max}(\vec{\pi}_k^{\max\{r_{k+1},i\}+p_{k+1}})$. Отсюда $C_{\max}(\pi_i^{'''}) - C_{\max}(\vec{\pi}^i_{k+1}) \leq p_{\max}(N)$, и неравенство (3.14) доказано. Пусть теперь неравенство (3.17) для некоторого $\bar{t} \leq i \leq P_{k+1}$ выполняется как строгое, т.е.

$$C_{\max}(\pi_i^{'''}) > C_{\max}(\pi^i_k). \tag{3.18}$$

Это значит, что $C_{j_{n_k}}(\pi_i^{'''}) > C_{j_{n_k}}(\pi^i_k)$ или, что то же самое, выполняется неравенство $\max\{C_{j_{n_k-1}}(\pi_i^{'''}), r_{j_{n_k}}\} + p_{j_{n_k}} > \max\{C_{j_{n_k-1}}(\pi^i_k), r_{j_{n_k}}\} + p_{j_{n_k}}$. Отсюда следуют неравенства

$$C_{j_{n_k-1}}(\pi_i^{'''}) > C_{j_{n_k-1}}(\pi^i_k) \tag{3.19}$$

и

$$C_{j_{n_k-1}}(\pi_i^{'''}) > r_{j_{n_k}}.$$

Неравенство (3.19) означает, что

$$\max\{C_{j_{n_k-2}}(\pi_i^{'''}), r_{j_{n_k-1}}\} + p_{j_{n_k-1}} > \max\{C_{j_{n_k-2}}(\pi^i_k), r_{j_{n_k-1}}\} + p_{j_{n_k-1}}.$$

Поэтому $C_{j_{n_k-2}}(\pi_i^{'''}) > C_{j_{n_k-2}}(\pi^i_k)$, $C_{j_{n_k-2}}(\pi_i^{'''}) > r_{j_{n_k-1}}$.
Продолжая аналогичные рассуждения, придем к заключению, что из (3.18) следует

$$C_{j_l}(\pi_i^{'''}) > C_{j_l}(\pi^i_k)\ \ \forall\ \ l = \overline{1, n_k} \tag{3.20}$$

и

$$C_{j_{l-1}}(\pi_i^{'''}) > r_{j_l}\ \ \forall\ \ l = \overline{2, n_k}. \tag{3.21}$$

Из (3.20) при $l = 1$ имеем $\max\{C_{k+1}(\pi_i^{'''}), r_{j_1}\} > \max\{r_{j_1}, i\}\ \forall\ i = \bar{t}, ..., P_{k+1}$, а значит, $C_{k+1}(\pi_i^{'''}) > r_{j_1}$. Отсюда и из (3.21) следует, что при расписании $\pi_i^{'''}$ требования обслуживаются без искусственных простоев прибора, т.е. $\pi_i^{'''} \in \vec{\Pi}_r(N_{k+1}, i)$. Таким образом, доказано равенство

$$C_{\max}(\pi_i^{'''}) = C_{\max}(\vec{\pi}_{k+1}^{i}), \tag{3.22}$$

а значит, и неравенство (3.14).

Далее, в процедуре h_0 (алгоритм 3.8) в качестве расписания $\pi_{k+1}^i \in \Pi(N_{k+1}, i)$ для всех $i = \bar{t}, ..., P_{k+1}$ выбирается

$$\begin{array}{lrcl} \text{либо} & \pi_i^{'} & = & (k+1, \pi_k^{\max\{r_{k+1}, i\}+p_{k+1}}) \in \Pi(N_{k+1}, i), \\ \text{либо} & \pi_i^{''} & = & (\pi_k^i, k+1) \in \Pi(N_{k+1}, i), \\ \text{либо} & \pi_i^{'''} & = & (k+1, \pi_k^i) \in \Pi(N_{k+1}, i). \end{array}$$

Пусть для некоторого $\bar{t} \le i \le P_{k+1}$ выполняется $\pi_{k+1}^i = \pi_i^{''}$. Покажем, что тогда

$$C_{\max}(\pi_i^{''}) - C_{\max}(\vec{\pi}_{k+1}^{i}) \le p_{\max}(N). \tag{3.23}$$

Согласно процедуре h_0 в качестве расписания π_{k+1}^i выбирается $\pi_i^{''}$ в следующих четырех случаях.

1. Пусть $L_{\max}(\pi_i^{'}) = L_{\max}(\pi_i^{''}) = L_{\max}(\pi_i^{'''}),\ C_{\max}(\pi_i^{''}) \le C_{\max}(\pi_i^{'}), C_{\max}(\pi_i^{''}) \le C_{\max}(\pi_i^{'''})$. Тогда с учётом (3.13) будем иметь

$$C_{\max}(\pi_i^{''}) - C_{\max}(\vec{\pi}_{k+1}^{i}) \le C_{\max}(\pi_i^{'}) - C_{\max}(\vec{\pi}_{k+1}^{i}) \le p_{\max}(N),$$

и неравенство (3.23) доказано.

2. Пусть $L_{\max}(\pi_i^{'}) = L_{\max}(\pi_i^{''}) < L_{\max}(\pi_i^{'''})$ и $C_{\max}(\pi_i^{''}) \le C_{\max}(\pi_i^{'})$. Тогда с учётом (3.13) $C_{\max}(\pi_i^{''}) - C_{\max}(\vec{\pi}_{k+1}^{i}) \le C_{\max}(\pi_i^{'}) - C_{\max}(\vec{\pi}_{k+1}^{i}) \le p_{\max}(N)$, и выполнимость (3.23) доказана.

3. Пусть $L_{\max}(\pi_i^{'''}) = L_{\max}(\pi_i^{''}) < L_{\max}(\pi_i^{'})$ и $C_{\max}(\pi_i^{''}) \le C_{\max}(\pi_i^{'''})$. Тогда с учётом (3.14) $C_{\max}(\pi_i^{''}) - C_{\max}(\vec{\pi}_{k+1}^{i}) \le C_{\max}(\pi_i^{'''}) - C_{\max}(\vec{\pi}_{k+1}^{i}) \le p_{\max}(N)$, и (3.23) справедливо.

4. Пусть $L_{\max}(\pi_i^{''}) < L_{\max}(\pi_i^{'})$ и

$$L_{\max}(\pi_i^{''}) < L_{\max}(\pi_i^{'''}). \tag{3.24}$$

Отметим, что

$$L_{\max}(\pi_i^{''}) = \max\left\{\max_{j \in N_k} L_j(\pi_i^{''}), L_{k+1}(\pi_i^{''})\right\}$$

и

$$C_{k+1}(\pi_i^{'''}) \le C_j(\pi_i^{'''}) \ \forall\ j \in N_k.$$

В силу (3.11) $d_{k+1} \ge d_j \ \forall\ j \in N_k$, поэтому $L_{\max}(\pi_i^{'''}) = \max\limits_{j\in N_k} L_j(\pi_i^{'''})$. Так как справедливо (3.24), то

$$\max\left\{\max_{j\in N_k} L_j(\pi_i^{''}), L_{k+1}(\pi_i^{''})\right\} < \max_{j\in N_k} L_j(\pi_i^{'''}). \tag{3.25}$$

Согласно алгоритму 1.5 найдутся такие $N^1, N^2 \subset N_k$ и соответствующие им частичные расписания π^1, π^2, что $\pi_k^i = (\pi^1, 1, \pi^2)$, $\pi^1 \in \Pi(N^1, i)$, $\pi^2 \in \Pi(N^2, C_{\max}(\pi^1, 1))$. Тогда $\pi_i^{''} = (\pi^1, 1, \pi^2, k{+}1)$, $\pi_i^{'''} = (k{+}1, \pi^1, 1, \pi^2)$. Согласно (3.11) $d_1 \le d_j$, $r_1 \ge r_j \ \forall\ j \in N_{k+1}$. Поэтому требование $1 \in J_{d_{\min}}(N_{k+1})$, и из леммы 1.5 следует существование таких требований $j^* \in J^*(\pi_i^{''})$ и $\bar{j}^* \in J^*(\pi_i^{'''})$, что $j^* \in N^2 \cup \{1\} \cup \{k+1\}$ и $\bar{j}^* \in N^2 \cup \{1\}$. Отсюда с учётом (3.25)

$$\max_{j\in N^2\cup\{1\}\cup\{k+1\}} L_j(\pi_i^{''}) < \max_{j\in N^2\cup\{1\}} L_j(\pi_i^{'''}). \tag{3.26}$$

Поскольку $r_1 \ge r_j \ \forall\ j \in N_{k+1}$, следовательно требования множеств $N^2 \cup \{1\} \cup \{k+1\}$ и $N^2 \cup \{1\}$ обслуживаются без искусственных простоев прибора при расписаниях $\pi_i^{''}$ и $\pi_i^{'''}$ соответственно, т.е.

$$C_l(\pi_i^{''}) \ge r_j, \quad C_l(\pi_i^{'''}) \ge r_j \tag{3.27}$$

для любых двух соседних требований $l, j \in N^2 \cup \{1\} \cup \{k+1\}$, $l \overset{\pi_i^{''}}{\to} j$, расписания $\pi_i^{''}$ и для любых двух соседних требований $l, j \in N^2 \cup \{1\}$, $l \overset{\pi_i^{'''}}{\to} j$, расписания $\pi_i^{'''}$. Тогда из (3.26) с учётом (3.27) следует, что $C_j(\pi_i^{''}) < C_j(\pi_i^{'''})$ $\forall\ j \in N^2 \cup \{1\}$. Это означает, что выполняется (3.18), из чего следует справедливость (3.22). В силу (3.27) получаем

$$C_{\max}(\pi_i^{''}) = C_{\max}(\pi_k^i) + p_{k+1}.$$

Отсюда, учитывая (3.18), (3.22), $C_{\max}(\pi_i^{''}) - C_{\max}(\vec{\pi}_{k+1}^i) = C_{\max}(\pi_k^i) + p_{k+1} - C_{\max}(\vec{\pi}_{k+1}^i) < C_{\max}(\pi_i^{'''}) + p_{k+1} - C_{\max}(\vec{\pi}_{k+1}^i) = p_{k+1} \le p_{\max}(N)$, и неравенство (3.23) доказано.

Поскольку в процедуре h_0 (алгоритм 3.8) в качестве расписания π_{k+1}^i для каждого $i = \bar{t}, ..., P_{k+1}$ выбирается одно из расписаний π_i', $\pi_i^{''}$, $\pi_i^{'''}$, то из (3.13), (3.14), (3.23) следует справедливость (3.12) при $m = k+1$.

Отметим, что $\pi_{h_0} = \pi_n^{\bar{t}} \in \Pi(N, \bar{t})$ и $\vec{\pi}_n^{\bar{t}} \in \vec{\Pi}_r(N, \bar{t})$. Поскольку $\bar{t} = \max\{r_n, t\}$, то $\Pi(N, \bar{t}) = \Pi(N, t)$ и $\vec{\Pi}_r(N, \bar{t}) = \vec{\Pi}_r(N, t)$, а значит, $\pi_{h_0} \in \Pi(N, t)$

и $C_{\max}(\vec{\pi}_n^{\bar{t}}) = C_{\max}(\vec{\pi}_r)$ для всех $\vec{\pi}_r \in \vec{\Pi}_r(N,t)$. Тогда из (3.12) при $i = \bar{t}$ следует справедливость утверждения леммы. Лемма доказана. □

Теперь покажем, что для задачи минимизации максимального временно́го смещения при ограничениях на параметры требований (1.63) ($r_i \le r_j \;\Rightarrow\; d_i \ge d_j \;\;\forall\;\; i,j \in N$) процедура h_0 строит приближённое расписание с оценкой абсолютной погрешности оптимального значения целевой функции, не превышающей $p_{\max}(N)$.

Теорема 3.4 *Пусть расписание $\pi_{h_0} \in \Pi(N,t)$ построено процедурой h_0, и выполняются условия* (1.63). *Тогда справедлива абсолютная погрешность $L_{\max}(\pi_{h_0}) - L_{\max}(\pi^*) \le p_{\max}(N)$ для любого $\pi^* \in \Pi^*(N,t)$.*

Доказательство. Покажем, что при всех $m = 1,...,n$ выполняется

$$L_{\max}(\pi_m^i) - L_{\max}(\pi_m^{*i}) \le p_{\max}(N) \;\;\forall\;\; i = \bar{t},...,P_m, \tag{3.28}$$

где расписание $\pi_m^i \in \Pi(N_m,i)$ построено процедурой h_0, $\pi_m^{*i} \in \Pi^*(N_m,i)$. Пусть $m = 1$. Очевидно, что $L_{\max}(\pi_1^i) = L_{\max}(\pi_1^{*i})$, где $\pi_1^i = (1) \in \Pi(N_1,i)$, $\pi_1^{*i} \in \Pi^*(N_1,i)$, для всех $i = \bar{t},...,P_1$, а значит, неравенства (3.28) выполняются. Пусть неравенства (3.28) выполняются при $m = k$, где $1 \le k \le n-1$. Докажем, что (3.28) справедливы при $m = k+1$. Тогда неравенства (3.28) будут доказаны для всех $m = 1,...,n$.

Согласно (3.11) $r_{k+1} = r_{\min}(N_{k+1})$, и выполняются (1.63), тогда из леммы 1.12 следует, что существует расписание $\pi_{k+1}^{*i} \in \Pi^*(N_{k+1},i)$, при котором справедливо $k+1 \overset{\pi_{k+1}^{*i}}{\to} j$ либо $j \overset{\pi_{k+1}^{*i}}{\to} k+1$ для всех $j \in N_k$, $i = \bar{t},...,P_{k+1}$.

1. Пусть $\bar{t} \le i \le P_{k+1}$ таково, что $\pi_{k+1}^{*i} = (k+1,\bar{\pi})$, где расписание $\bar{\pi} \in \Pi(N_k, \max\{r_{k+1},i\} + p_{k+1})$.

Покажем, что справедливо неравенство

$$L_{\max}(\pi_i') - L_{\max}(\pi_{k+1}^{*i}) \le p_{\max}(N), \tag{3.29}$$

где $\pi_i' = (k+1, \pi_k^{\max\{r_{k+1},i\}+p_{k+1}}) \in \Pi(N_{k+1},i)$.

Согласно (3.11) $d_{k+1} \ge d_j \;\forall\; j \in N_k$. Кроме того, $C_{k+1}(\pi_i') \le C_j(\pi_i')$ и $C_{k+1}(\pi_{k+1}^{*i}) \le C_j(\pi_{k+1}^{*i})$ для любого $j \in N_k$. Отсюда

$$L_{\max}(\pi_i') = L_{\max}(\pi_k^{\max\{r_{k+1},i\}+p_{k+1}}),\; L_{\max}(\pi_{k+1}^{*i}) = L_{\max}(\bar{\pi}),$$

а значит, $L_{\max}(\pi_i')-L_{\max}(\pi_{k+1}^{*i}) = L_{\max}(\pi_k^{\max\{r_{k+1},i\}+p_{k+1}})-L_{\max}(\bar{\pi})$. Так как $\bar{\pi} \in \Pi(N_k, \max\{r_{k+1}, i\}+p_{k+1})$ и $\pi_k^{*\max\{r_{k+1},i\}+p_{k+1}} \in \Pi^*(N_k, \max\{r_{k+1}, i\}+p_{k+1})$, то $L_{\max}(\bar{\pi}) \geq L_{\max}(\pi_k^{*\max\{r_{k+1},i\}+p_{k+1}})$. Отсюда с учётом (3.28) при $m = k$ следует

$$L_{\max}(\pi_i') - L_{\max}(\pi_{k+1}^{*i}) \leq L_{\max}(\pi_k^{\max\{r_{k+1},i\}+p_{k+1}}) - L_{\max}(\pi_k^{*\max\{r_{k+1},i\}+p_{k+1}}) \leq p_{\max}(N),$$

и неравенство (3.29) доказано.

Далее, в процедуре h_0 (алгоритм 3.8) в качестве расписания $\pi_{k+1}^i \in \Pi(N_{k+1}, i)\ \forall\, i = \bar{t}, ..., P_{k+1}$ выбирается $\pi_i' = (k+1, \pi_k^{\max\{r_{k+1},i\}+p_{k+1}}) \in \Pi(N_{k+1}, i)$, либо $\pi_i'' = (\pi_k^i, k+1) \in \Pi(N_{k+1}, i)$, либо $\pi_i''' = (k+1, \pi_k^i) \in \Pi(N_{k+1}, i)$. Пусть для выбранного $\bar{t} \leq i \leq P_{k+1}$ выполняется $\pi_{k+1}^i = \pi_i''$. Покажем, что

$$L_{\max}(\pi_i'') - L_{\max}(\pi_{k+1}^{*i}) \leq p_{\max}(N). \tag{3.30}$$

Так как $L_{\max}(\pi_i'') \leq L_{\max}(\pi_i')$, то

$$L_{\max}(\pi_i'') - L_{\max}(\pi_{k+1}^{*i}) \leq L_{\max}(\pi_i') - L_{\max}(\pi_{k+1}^{*i}).$$

Отсюда с учётом (3.29) следует (3.30).
Пусть теперь выполняется $\pi_{k+1}^i = \pi_i'''$. Покажем ,что

$$L_{\max}(\pi_i''') - L_{\max}(\pi_{k+1}^{*i}) \leq p_{\max}(N). \tag{3.31}$$

Так как $L_{\max}(\pi_i''') \leq L_{\max}(\pi_i')$, то

$$L_{\max}(\pi_i''') - L_{\max}(\pi_{k+1}^{*i}) \leq L_{\max}(\pi_i') - L_{\max}(\pi_{k+1}^{*i}).$$

Отсюда с учётом (3.29) следует (3.31).

Таким образом, для тех $\bar{t} \leq i \leq P_{k+1}$, при которых $k+1 \overset{\pi_{k+1}^{*i}}{\to} j\ \forall\ j \in N_k$ неравенства (3.28) доказаны при $m = k+1$.

2. Пусть теперь для некоторого $\bar{t} \leq i \leq P_{k+1}$ выполняется $\pi_{k+1}^{*i} = (\bar{\pi}, k+1)$, где $\bar{\pi} \in \Pi(N_k, i)$.

Покажем справедливость (3.30) для выбранного $\bar{t} \leq i \leq P_{k+1}$. Отметим, что

$$L_{\max}(\pi_i'')-L_{\max}(\pi_{k+1}^{*i}) = \max\{L_{\max}(\pi_k^i), L_{k+1}(\pi_i'')\}-\max\{L_{\max}(\bar{\pi}), L_{k+1}(\pi_{k+1}^{*i})\}.$$

а) Пусть $L_{\max}(\pi_k^i) \geq L_{k+1}(\pi_i'')$. Тогда

$$L_{\max}(\pi_i^{''}) - L_{\max}(\pi_{k+1}^{*i}) = L_{\max}(\pi_k^i) - \max\{L_{\max}(\bar{\pi}), L_{k+1}(\pi_{k+1}^{*i})\}.$$

Если $L_{\max}(\bar{\pi}) \geq L_{k+1}(\pi_{k+1}^{*i})$, то $L_{\max}(\pi_i^{''}) - L_{\max}(\pi_{k+1}^{*i}) = L_{\max}(\pi_k^i) - L_{\max}(\bar{\pi})$. Поскольку $\bar{\pi} \in \Pi(N_k, i)$ и $\pi_k^{*i} \in \Pi^*(N_k, i)$, то $L_{\max}(\bar{\pi}) \geq L_{\max}(\pi_k^{*i})$. Отсюда с учётом (3.28) при $m = k$ имеем

$$L_{\max}(\pi_k^i) - L_{\max}(\bar{\pi}) \leq L_{\max}(\pi_k^i) - L_{\max}(\pi_k^{*i}) \leq p_{\max}(N), \tag{3.32}$$

и $L_{\max}(\pi_i^{''}) - L_{\max}(\pi_{k+1}^{*i}) \leq p_{\max}(N)$. Таким образом, неравенство (3.30) в этом случае доказано. Если $L_{\max}(\bar{\pi}) < L_{k+1}(\pi_{k+1}^{*i})$, то $L_{\max}(\pi_i^{''}) - L_{\max}(\pi_{k+1}^{*i}) = L_{\max}(\pi_k^i) - L_{k+1}(\pi_{k+1}^{*i}) < L_{\max}(\pi_k^i) - L_{\max}(\bar{\pi})$. Тогда из (3.32) следует справедливость (3.30).

b) Пусть $L_{\max}(\pi_k^i) < L_{k+1}(\pi_i^{''})$. Тогда

$$L_{\max}(\pi_i^{''}) - L_{\max}(\pi_{k+1}^{*i}) = L_{k+1}(\pi_i^{''}) - \max\{L_{\max}(\bar{\pi}), L_{k+1}(\pi_{k+1}^{*i})\}.$$

Если $L_{k+1}(\pi_{k+1}^{*i}) \geq L_{\max}(\bar{\pi})$, то $L_{\max}(\pi_i^{''}) - L_{\max}(\pi_{k+1}^{*i}) = L_{k+1}(\pi_i^{''}) - L_{k+1}(\pi_{k+1}^{*i}) = C_{\max}(\pi_i^{''}) - C_{\max}(\pi_{k+1}^{*i})$. Согласно (3.23) $L_{\max}(\pi_i^{''}) - L_{\max}(\pi_{k+1}^{*i}) \leq C_{\max}(\vec{\pi}_{k+1}^i) + p_{\max}(N) - C_{\max}(\pi_{k+1}^{*i})$. Поскольку $\vec{\pi}_{k+1}^i, \pi_{k+1}^{*i} \in \Pi(N_{k+1}, i)$ и $\vec{\pi}_{k+1}^i \in \vec{\Pi}_r(N_{k+1}, i)$, то $C_{\max}(\vec{\pi}_{k+1}^i) \leq C_{\max}(\pi_{k+1}^{*i})$. Отсюда

$$L_{k+1}(\pi_i^{''}) - L_{k+1}(\pi_{k+1}^{*i}) \leq p_{\max}(N) \tag{3.33}$$

и $L_{\max}(\pi_i^{''}) - L_{\max}(\pi_{k+1}^{*i}) \leq p_{\max}(N)$. Таким образом, неравенство (3.30) в этом случае доказано. Если $L_{k+1}(\pi_{k+1}^{*i}) < L_{\max}(\bar{\pi})$, то $L_{\max}(\pi_i^{''}) - L_{\max}(\pi_{k+1}^{*i}) = L_{k+1}(\pi_i^{''}) - L_{\max}(\bar{\pi}) < L_{k+1}(\pi_i^{''}) - L_{k+1}(\pi_{k+1}^{*i})$ Тогда из (3.33) следует справедливость (3.30).

Пусть для выбранного $\bar{t} \leq i \leq P_{k+1}$ выполняется $\pi_{k+1}^i = \pi_i^{'}$. Покажем, что для этого i справедливо (3.29). Согласно предположению $L_{\max}(\pi_i^{'}) \leq L_{\max}(\pi_i^{''})$, а поскольку (3.30) выполняется для выбранного i, то $L_{\max}(\pi_i^{'}) - L_{\max}(\pi_{k+1}^{*i}) \leq L_{\max}(\pi_i^{''}) - L_{\max}(\pi_{k+1}^{*i}) \leq p_{\max}(N)$, и неравенство (3.29) доказано.

Пусть теперь выполняется $\pi_{k+1}^i = \pi_i^{'''}$. Покажем, что справедливо (3.31). Согласно предположению $L_{\max}(\pi_i^{'''}) \leq L_{\max}(\pi_i^{''})$, а так как (3.30) выполняется для выбранного i, то $L_{\max}(\pi_i^{'''}) - L_{\max}(\pi_{k+1}^{*i}) \leq L_{\max}(\pi_i^{''}) - L_{\max}(\pi_{k+1}^{*i}) \leq p_{\max}(N)$, и неравенство (3.31) доказано.

Поскольку в процедуре h_0 (алгоритм 3.8) в качестве расписания π_{k+1}^i для каждого $i = \bar{t}, ..., P_{k+1}$ выбирается одно из расписаний $\pi_i^{'}$, $\pi_i^{''}$, $\pi_i^{'''}$, то из (3.29), (3.30), (3.31) следует справедливость (3.28) при $m = k + 1$.

Отметим, что $\pi_{h_0} = \pi_n^{\bar{t}} \in \Pi(N, \bar{t})$ и $\pi_{k+1}^{*i} \in \Pi^*(N, \bar{t})$. Поскольку $\bar{t} = \max\{r_n, t\}$, то $\Pi(N, \bar{t}) = \Pi(N, t)$ и $\Pi^*(N, \bar{t}) = \Pi^*(N, t)$, а значит, $\pi_{h_0} \in \Pi(N, t)$

и $L_{\max}(\pi^{*i}_{k+1}) = L_{\max}(\pi^*)$ для всех $\pi^* \in \Pi^*(N,t)$. Тогда из (3.28) при $i = \bar{t}$ следует справедливость утверждения теоремы. Теорема доказана. □

Теорема 3.5 *Трудоёмкость процедуры h_0 составляет $O(n^2P)$ операций, где*

$$P = \max\{r_{\max}(N), t\} + \sum_{j=1}^{n} p_j - \max\{r_{\min}(N), t\}.$$

Доказательство. Для перенумерации требований согласно (3.11) достаточно $O(n \log n)$ операций [11]. На k-й итерации процедуры количество строящихся расписаний π^i_{k+1} не превышает P. Для вычисления значений $L_{\max}(\pi'_i)$, $L_{\max}(\pi''_i)$, $L_{\max}(\pi'''_i)$, $C_{\max}(\pi'_i)$, $C_{\max}(\pi''_i)$, $C_{\max}(\pi'''_i)$ требуется $O(n)$ операций. Поэтому для построения каждого расписания π^i_{k+1} достаточно $O(n)$ операций, а для построения на k-й итерации всех расписаний π^i_{k+1} потребуется $O(nP)$ операций. Следовательно, трудоёмкость одной итерации – $O(nP)$ операций, а поскольку процедура h_0 состоит из n итераций, то её трудоёмкость составляет $O(n^2P)$ операций. Теорема доказана. □

Таким образом, процедурой h_0 за $O(n^2P)$ операций строится приближённое расписание как для задачи на быстродействие, так и для задачи минимизации максимального временно́го смещения с оценкой абсолютной погрешности оптимального значения целевой функции ($C_{\max}$ и $L_{\max}$), не превышающей $p_{\max}(N)$.

Необходимо заметить, что алгоритм Шраге тоже строит приближенное решение с оценкой абсолютной погрешности целевой функции ($L_{\max}$) не превышающей $p_{\max}(N)$ за количество операций не более $O(n^2)$.

Также следует отметить, что условиям (1.63) ($r_i \le r_j => d_i \ge d_j, \forall_i, j \in N$) соответствует случай, для которого доказана NP—трудность в сильном смысле задачи $1|r_j|L_{\max}$ [95].

Таким образом, для NP—трудной в сильном смысле задачи построен псевдономинальный алгоритм, который находит приближенное решение с гарантированной абсолютной погрешностью целевых функций $C_{\max}$ и $L_{\max}$.

3.5.2 Процедура построения допустимого расписания для задачи $1|r_i \le r_j, d_i \ge d_j|L_{\max}$

В этом параграфе для NP—трудного случая задачи $1|r_j|L_{\max}$ при ограничениях на параметры требований (1.63) ($r_i \le r_j \;\Rightarrow\; d_i \ge d_j \;\; \forall \;\; i,j \in N$)

предлагается процедура, которая строит расписание со значением целевой функции, не превышающей заданной величины y либо устанавливает, что такого расписания не существует. Построенное этой процедурой расписание является оптимальным по быстродействию среди всех расписаний допустимых относительно значения y.

Опишем ниже процедуру h построения расписания $\pi_h \in \Pi(N,t)$, оптимального по быстродействию среди всех расписаний со значением целевой функции не превышающим y, т.е. $C_{\max}(\pi_h) = \min\{C_{\max}(\pi) | \pi \in \Pi(N,t), L_{\max}(\pi) \leq y\}$, y – любое вещественное число, если такое расписание существует.

Процедура состоит из n итераций. Внутри итерации для каждого целочисленного момента времени, число которых не превосходит P, строится допустимое расписание путём добавления к уже построенному расписанию требования из множества неупорядоченных с максимальным моментом поступления и минимальным директивным сроком.

Алгоритм 3.9 Процедура h.

Первоначально перенумеруем требования множества N таким образом, чтобы соблюдались неравенства (3.11). Полагаем $\bar{t} = \max\{r_n, t\}$, $N_1 = \{1\}$, $P_1 = \max\{r_1, t\} + \sum\limits_{j \in N} p_j - p_1$,

$\pi_1^i = \emptyset$, *если* $\max\{r_1, i\} + p_1 - d_1 > y$, *и*
$\pi_1^i = (1)$, *если* $\max\{r_1, i\} + p_1 - d_1 \leq y \; \forall \; i = \overline{\bar{t}, P_1}$.
Пусть уже известны N_k, P_k, π_k^i *для всех* $i = \overline{\bar{t}, P_k}$, *и* $1 \leq k < n$.
Полагаем $N_{k+1} = N_k \cup \{k+1\}$, $P_{k+1} = P_k - p_{k+1}$. *Далее, для каждого* $i = \bar{t}, ..., P_{k+1}$ *строим расписания* $\pi_i', \pi_i'', \pi_{k+1}^i \in \Pi(N_{k+1}, i)$ *следующим образом.*
Полагаем
$\pi_i' = \emptyset$, *если* $L_{\max}(k+1, \pi_k^{\max\{r_{k+1}, i\} + p_{k+1}}) > y$, *и* $\pi_i' = (k+1, \pi_k^{\max\{r_{k+1}, i\} + p_{k+1}})$ *в противном случае,*
$\pi_i'' = \emptyset$, *если* $L_{\max}(\pi_k^i, k+1) > y$, *и* $\pi_i'' = (\pi_k^i, k+1)$ *в противном случае.*
Полагаем
$\Pi_i = \{\pi \in \{\pi_i', \pi_i''\} : \pi \neq \emptyset\}$, $\pi_{k+1}^i = \emptyset$, *если* $\Pi_i = \emptyset$, *и*
$\pi_{k+1}^i = \arg\min\{C_{\max}(\pi) | \pi \in \Pi_i\}$, *если* $\Pi_i \neq \emptyset$.
При $k = n$ *полагаем* $\pi_h = \pi_k^{\bar{t}}$, *и процесс заканчивается.*

Очевидно, что в процедуре h (алгоритм 3.9) перенумерация требований множества N согласно (3.11) возможна в силу условий (1.63) ($r_i \leq r_j \Rightarrow d_i \geq d_j \; \forall \, i, j \in N$). В случае, когда (1.63) не выполняется, то если с помощью процедуры h было построено расписание, то оно является оптимальным по быстродействию среди допустимых $L_{\max}(\pi) \leq y$. В случае, когда на выходе процедуры будет пустое расписание, то это не исключает существование допустимого расписания.

Поскольку $r_{k+1} \leq r_1$ и $\bar{t} \leq i \leq P_k$, то $\max\{r_{k+1}, i\} + p_{k+1} \leq P_k$. Кроме

того, $P_{k+1} \le P_k$. Поэтому расписания $\pi_k^{\max\{r_{k+1},i\}+p_{k+1}}$ и π_k^i, $i = \overline{\bar{t}, P_{k+1}}$, которые используются для построения π_{k+1}^i, $k = 1, ..., n-1$, будут уже построены на предыдущей итерации процедуры.

Теорема 3.6 *Пусть расписание $\pi_h \in \Pi(N,t)$ построено процедурой h (алгоритм 3.9), y – любое вещественное число и выполняются условия* (1.63) $(r_i \le r_j \;\Rightarrow\; d_i \ge d_j \;\;\forall \;\; i,j \in N)$. *Если расписание $\pi_h \ne \pi^\emptyset$ (не пустое расписание), то выполняется $C_{\max}(\pi_h) = \min\{C_{\max}(\pi) | \pi \in \Pi(N,t), L_{\max}(\pi) \le y\}$, в противном случае $L_{\max}(\pi) > y$ для всех $\pi \in \Pi(N,t)$.*

Доказательство. Покажем, что для любой пары номеров m, i таких, что $1 \le m \le n$, $\bar{t} \le i \le P_m$, выполняется равенство

$$C_{\max}(\pi_m^i) = \min\{C_{\max}(\pi) | \pi \in \Pi(N_m, i), L_{\max}(\pi) \le y\}, \tag{3.34}$$

если $\pi_m^i \ne \emptyset$, и

$$L_{\max}(\pi) > y \;\; \forall \;\; \pi \in \Pi(N_m, i), \tag{3.35}$$

если $\pi_m^i = \emptyset$, где расписание $\pi_m^i \in \Pi(N_m, i)$ построено процедурой h.

Пусть $m = 1$, а номер i таков, что $\bar{t} \le i \le P_1$. Тогда $|\Pi(N_1, i)| = 1$ и $\pi = \pi_i' = \pi_i'' = (1)$, $\pi \in \Pi(N_1, i)$. Если $\pi_1^i \ne \emptyset$, то $\Pi_i \ne \emptyset$ и, очевидно, что (3.34) выполняется. Если $\pi_1^i = \emptyset$, то $\Pi_i = \emptyset$ и, следовательно, (3.35) выполняется. Пусть соотношения (3.34), (3.35) выполняются при $m = k$, где $1 \le k \le n-1$, и для всех $i = \bar{t}, ..., P_k$. Покажем, что тогда (3.34), (3.35) справедливы при $m = k+1$ и любом $i = \bar{t}, ..., P_{k+1}$. Тем самым соотношения (3.34), (3.35) будут доказаны для любых $m = 1, ..., n$, $i = \bar{t}, ..., P_m$.

Итак, пусть номер i таков, что $\bar{t} \le i \le P_{k+1}$. Выберем произвольно $\pi \in \Pi(N_{k+1}, i)$. Согласно алгоритму 1.5 найдутся такие $N^1, N^2 \subset N_{k+1}$, $N_1 \cap N_2 = \emptyset$, $N_1 \cup N_2 = N_{k+1}$, и соответствующие им частичные расписания π^1, π^2, что $\pi = (\pi^1, 1, \pi^2)$, $\pi^1 \in \Pi(N^1, i)$, $\pi^2 \in \Pi(N^2, C_{\max}(\pi^1, 1))$. Заметим, что $k+1 \in N^1$ либо $k+1 \in N^2$.

Рассмотрим какому множеству (N^1 или N^2) принадлежит $k+1$.

1. Пусть $k+1 \in N^1$. Покажем, что из

$$L_{\max}(\pi_i') \le y, \;\; L_{\max}(\pi) \le y \tag{3.36}$$

следует

$$C_{\max}(\pi_i') \le C_{\max}(\pi), \tag{3.37}$$

а из

$$L_{\max}(\pi_i') > y \tag{3.38}$$

следует

$$L_{\max}(\pi) > y, \tag{3.39}$$

где $\pi'_i = (k+1, \pi_k^{\max\{r_{k+1},i\}+p_{k+1}}) \in \Pi(N_{k+1}, i)$.

Согласно алгоритму 1.5 найдутся такие $\tilde{N}^1, \tilde{\tilde{N}}^1 \subset N^1$ и соответствующие им частичные расписания $\tilde{\pi}^1$, $\tilde{\tilde{\pi}}^1$, что $\pi^1 = (\tilde{\pi}^1, k+1, \tilde{\tilde{\pi}}^1)$, $\tilde{\pi}^1 \in \Pi(\tilde{N}^1, i)$, $\tilde{\tilde{\pi}}^1 \in \Pi(\tilde{\tilde{N}}^1, C_{\max}(\tilde{\pi}^1, k+1))$. Тогда

$$\pi = (\tilde{\pi}^1, k+1, \tilde{\tilde{\pi}}^1, 1, \pi^2).$$

Построим расписание

$$\bar{\pi} = (k+1, \tilde{\pi}^1, \tilde{\tilde{\pi}}^1, 1, \pi^2),$$

отличающееся от расписания π порядком обслуживания требования $k+1$. Очевидно, что

$$C_{\max}(\pi'_i) = C_{\max}(\pi_k^{\max\{r_{k+1},i\}+p_{k+1}}) \tag{3.40}$$

и

$$C_{\max}(\bar{\pi}) = C_{\max}(\tilde{\pi}^1, \tilde{\tilde{\pi}}^1, 1, \pi^2). \tag{3.41}$$

Согласно (3.11) $d_{k+1} \geq d_j$, $r_{k+1} \leq r_j \ \forall\ j \in N_k$. Кроме того, $C_{k+1}(\pi'_i) \leq C_j(\pi'_i)$ и $C_{k+1}(\bar{\pi}) \leq C_j(\bar{\pi}) \ \forall\ j \in N_k$. Отсюда

$$L_{\max}(\pi'_i) = L_{\max}(\pi_k^{\max\{r_{k+1},i\}+p_{k+1}}) \tag{3.42}$$

и

$$L_{\max}(\bar{\pi}) = L_{\max}(\tilde{\pi}^1, \tilde{\tilde{\pi}}^1, 1, \pi^2). \tag{3.43}$$

Поскольку порядок обслуживания требований множества $\tilde{N}^1 \cup \tilde{\tilde{N}}^1 \cup N^2 \cup \{1\}$ одинаков при расписаниях π и $\bar{\pi}$ и $r_{k+1} \leq r_j \ \forall\ j \in N_k$, то

$$C_j(\pi) \geq C_j(\bar{\pi}) \ \ \forall\ \ j \in \tilde{N}^1 \cup \tilde{\tilde{N}}^1 \cup N^2 \cup \{1\}. \tag{3.44}$$

В силу (3.11) имеем $d_1 \leq d_j$, $r_1 \geq r_j \ \forall\ j \in N_{k+1}$, поэтому требование $1 \in J_{d_{\min}}(N_{k+1})$. Тогда из леммы 1.5 следует существование таких требований $j^* \in J^*(\pi)$ и $\bar{j}^* \in J^*(\bar{\pi})$, что $j^*, \bar{j}^* \in N^2 \cup \{1\}$. Отсюда с учётом (3.44) $L_{\max}(\pi) - L_{\max}(\bar{\pi}) = \max\limits_{j \in N^2 \cup \{1\}} L_j(\pi) - \max\limits_{j \in N^2 \cup \{1\}} L_j(\bar{\pi}) \geq 0$. Следовательно,

$$L_{\max}(\pi) \geq L_{\max}(\bar{\pi}). \tag{3.45}$$

Пусть справедливо (3.36). Тогда с учётом (3.42), (3.43), (3.45)

$$L_{\max}(\pi_k^{\max\{r_{k+1},i\}+p_{k+1}}) \le y,\ L_{\max}(\tilde{\pi}^1,\tilde{\tilde{\pi}}^1,1,\pi^2) \le y.$$

Кроме того,

$$\pi_k^{\max\{r_{k+1},i\}+p_{k+1}},(\tilde{\pi}^1,\tilde{\tilde{\pi}}^1,1,\pi^2) \in \Pi(N_k,\max\{r_{k+1},i\}+p_{k+1}).$$

Следовательно, в силу (3.34) при $m=k$, выполняется неравенство

$$C_{\max}(\pi_k^{\max\{r_{k+1},i\}+p_{k+1}}) \le C_{\max}(\tilde{\pi}^1,\tilde{\tilde{\pi}}^1,1,\pi^2).$$

Отсюда с учётом (3.44), (3.41), (3.40)

$$C_{\max}(\pi) \ge C_{\max}(\bar{\pi}) = C_{\max}(\tilde{\pi}^1,\tilde{\tilde{\pi}}^1,1,\pi^2) \ge C_{\max}(\pi_k^{\max\{r_{k+1},i\}+p_{k+1}}) = C_{\max}(\pi_i'),$$

и неравенство (3.37) доказано.

Пусть справедливо (3.38). Тогда в силу (3.42)

$$L_{\max}(\pi_k^{\max\{r_{k+1},i\}+p_{k+1}}) > y.$$

Из (3.34) при $m=k$ следует, что

$$L_{\max}(\hat{\pi}) > y\ \forall\ \hat{\pi} \in \Pi(N_k,\max\{r_{k+1},i\}+p_{k+1}),$$

а так как $(\tilde{\pi}^1,\tilde{\tilde{\pi}}^1,1,\pi^2) \in \Pi(N_k,\max\{r_{k+1},i\}+p_{k+1})$, то $L_{\max}(\tilde{\pi}^1,\tilde{\tilde{\pi}}^1,1,\pi^2) > y$. Отсюда с учётом (3.45), (3.43) $L_{\max}(\pi) \ge L_{\max}(\bar{\pi}) = L_{\max}(\tilde{\pi}^1,\tilde{\tilde{\pi}}^1,1,\pi^2) > y$, и неравенство (3.39) доказано.

2. Пусть теперь $k+1 \in N^2$. Покажем, что из

$$L_{\max}(\pi_i'') \le y,\ \ L_{\max}(\pi) \le y \tag{3.46}$$

следует

$$C_{\max}(\pi_i'') \le C_{\max}(\pi), \tag{3.47}$$

а из

$$L_{\max}(\pi_i'') > y \tag{3.48}$$

следует (3.39), где $\pi_i'' = (\pi_k^i, k+1) \in \Pi(N_{k+1},i)$.

Согласно алгоритму 1.5 найдутся такие $\tilde{N}^2, \tilde{\tilde{N}}^2 \subset N^2$ и соответствующие им частичные расписания $\tilde{\pi}^2$, $\tilde{\tilde{\pi}}^2$, что $\pi^2 = (\tilde{\pi}^2, k+1,\ \tilde{\tilde{\pi}}^2)$, $\tilde{\pi}^2 \in \Pi(\tilde{N}^2, C_{\max}(\pi^1,1))$, $\tilde{\tilde{\pi}}^2 \in \Pi(\tilde{\tilde{N}}^2, C_{\max}(\pi^1,1,\tilde{\pi}^2,k+1))$. Тогда

$$\pi = (\pi^1,1,\tilde{\pi}^2,k+1,\tilde{\tilde{\pi}}^2).$$

Построим расписание

$$\bar{\pi} = (\pi^1, 1, \tilde{\pi}^2, \tilde{\tilde{\pi}}^2, k+1),$$

отличающееся от расписания π порядком обслуживания требования $k+1$. Согласно (3.11), $r_1 \geq r_j$, $r_{k+1} \leq r_j \ \forall \ j \in N_{k+1}$. Следовательно, при расписаниях π, $\bar{\pi}$ требования множества $N^2 \cup \{1\}$ обслуживаются без простоев прибора, а при расписании $\pi_i^{''}$ между требованиями множества N_k и $(k+1)$ простой прибор также отсутствует. Отсюда

$$C_{\max}(\pi) = C_{\max}(\bar{\pi}) = C_{\max}(\pi^1, 1, \tilde{\pi}^2, \tilde{\tilde{\pi}}^2) + p_{k+1}, \tag{3.49}$$

и

$$C_{\max}(\pi_i^{''}) = C_{\max}(\pi_k^i) + p_{k+1}. \tag{3.50}$$

Отметим, что

$$L_{\max}(\pi_i^{''}) = \max\{L_{\max}(\pi_k^i), L_{k+1}(\pi_i^{''})\}, \tag{3.51}$$

и

$$L_{\max}(\bar{\pi}) = \max\{L_{\max}(\pi^1, 1, \tilde{\pi}^2, \tilde{\tilde{\pi}}^2), L_{k+1}(\bar{\pi})\}. \tag{3.52}$$

Покажем, что выполняется (3.45). В силу (3.11), $d_{k+1} \geq d_j \ \forall \ j \in N_k$. Кроме того, $C_{k+1}(\pi) \leq C_j(\pi) \ \forall \ j \in \tilde{N}^2$ и, следовательно, $L_{k+1}(\pi) \leq L_j(\pi) \ \forall \ j \in \tilde{\tilde{N}}^2$. Поэтому $L_{\max}(\pi) = \max\limits_{j \in N_k} L_j(\pi)$. Отсюда

$$L_{\max}(\pi) - L_{\max}(\bar{\pi}) = \max_{j \in N_k} L_j(\pi) - \max\left\{\max_{j \in N_k} L_j(\bar{\pi}), L_{k+1}(\bar{\pi})\right\}.$$

Пусть $\max\limits_{j \in N_k} L_j(\bar{\pi}) \geq L_{k+1}(\bar{\pi})$, т.е.

$$L_{\max}(\pi) - L_{\max}(\bar{\pi}) = \max_{j \in N_k} L_j(\pi) - \max_{j \in N_k} L_j(\bar{\pi}).$$

Поскольку $C_j(\pi) \geq C_j(\bar{\pi})$ и, следовательно, $L_j(\pi) \geq L_j(\bar{\pi}) \ \forall \ j \in N_k$, то (3.45) выполняется.

Пусть $\max\limits_{j \in N_k} L_j(\bar{\pi}) < L_{k+1}(\bar{\pi})$, т.е.

$$L_{\max}(\pi) - L_{\max}(\bar{\pi}) = \max_{j \in N_k} L_j(\pi) - L_{k+1}(\bar{\pi}).$$

Так как $d_{k+1} \geq d_j \ \forall \ j \in N_k$, то с учётом (3.49) получим

$$C_{\max}(\pi) - d_j \geq C_{\max}(\bar{\pi}) - d_{k+1} \ \forall \ j \in N_k.$$

Это означает, что временно́е смещение требования, обслуживаемого на последнем месте при расписании π, не меньше, чем $L_{k+1}(\bar{\pi})$, и неравенство (3.45) доказано.

Пусть справедливо (3.46). Тогда с учётом (3.51), (3.52) имеем

$$L_{\max}(\pi_k^i) \le y \text{ и } L_{\max}(\pi^1, 1, \tilde{\pi}^2, \tilde{\tilde{\pi}}^2) \le y.$$

Кроме того, $\pi_k^i, (\pi^1, 1, \tilde{\pi}^2, \tilde{\tilde{\pi}}^2) \in \Pi(N_k, i)$. Следовательно, учитывая (3.34) при $m = k$,

$$C_{\max}(\pi_k^i) \le C_{\max}(\pi^1, 1, \tilde{\pi}^2, \tilde{\tilde{\pi}}^2).$$

Отсюда с учётом (3.50), (3.49), будем иметь

$$C_{\max}(\pi_i'') = C_{\max}(\pi_k^i) + p_{k+1} \le C_{\max}(\tilde{\pi}^1, \tilde{\tilde{\pi}}^1, 1, \pi^2) + p_{k+1} = C_{\max}(\bar{\pi}) = C_{\max}(\pi),$$

и неравенство (3.47) доказано.

Пусть справедливо (3.48). Тогда либо $L_{\max}(\pi_k^i) > y$, либо $L_{\max}(\pi_k^i) \le y$ и $L_{k+1}(\pi_i'') > y$. Пусть $L_{\max}(\pi_k^i) > y$. Из (3.34) при $m = k$ следует, что

$$L_{\max}(\hat{\pi}) > y \ \forall \ \hat{\pi} \in \Pi(N_k, i),$$

а так как $(\pi^1, 1, \tilde{\pi}^2, \tilde{\tilde{\pi}}^2) \in \Pi(N_k, i)$, то

$$L_{\max}(\pi^1, 1, \tilde{\pi}^2, \tilde{\tilde{\pi}}^2) > y$$

и в силу (3.52) получим $L_{\max}(\bar{\pi}) > y$. Отсюда с учётом (3.45), следует неравенство (3.39). Пусть теперь

$$L_{\max}(\pi_k^i) \le y \text{ и } L_{k+1}(\pi_i'') > y.$$

Так как $\pi_k^i, (\pi^1, 1, \tilde{\pi}^2, \tilde{\tilde{\pi}}^2) \in \Pi(N_k, i)$, то из (3.34) при $m = k$ следует, что $C_{\max}(\pi_k^i) \le C_{\max}(\pi^1, 1, \tilde{\pi}^2, \tilde{\tilde{\pi}}^2)$. Отсюда с учётом (3.50), (3.49), имеем

$$C_{\max}(\pi_i'') = C_{\max}(\pi_k^i) + p_{k+1} \le C_{\max}(\tilde{\pi}^1, \tilde{\tilde{\pi}}^1, 1, \pi^2) + p_{k+1} = C_{\max}(\bar{\pi}),$$

а значит, $L_{k+1}(\pi_i'') \le L_{k+1}(\bar{\pi})$ и $L_{k+1}(\bar{\pi}) > y$. Тогда из (3.45), (3.52) следует (3.39).

Далее, пусть $\pi_{k+1}^i \ne \pi^\emptyset$. Тогда $\Pi_i \ne \emptyset$. Это возможно в следующих трех случаях.

1. Пусть $L_{\max}(\pi_i') \le y$, $L_{\max}(\pi_i'') \le y$. Из (3.36), (3.46) следует, что для любого расписания $\pi \in \Pi(N_{k+1}, i)$, для которого $L_{\max}(\pi) \le y$ справедливо (3.37) либо (3.47). Значит, (3.34) при $m = k + 1$ доказано.

2. Пусть $L_{\max}(\pi_i') \le y$, $L_{\max}(\pi_i'') > y$. Из (3.36) следует, что для любого расписания $\pi = (\pi^1, 1, \pi^2) \in \Pi(N_{k+1}, i)$, для которого $L_{\max}(\pi) \le y$, в случае $k + 1 \in N^1$ справедливо (3.37). Кроме того, из (3.48) следует (3.39) для всех

$\pi \in \Pi(N_{k+1}, i)$, если $k+1 \in N^2$. Поскольку $(k+1) \in N^1$ либо $k+1 \in N^2$, то (3.34) при $m = k+1$ доказано в этом случае.

3. Пусть $L_{\max}(\pi_i') > y$, $L_{\max}(\pi_i'') \le y$. Из (3.46) следует, что для любого расписания $\pi = (\pi^1, 1, \pi^2) \in \Pi(N_{k+1}, i)$, для которого $L_{\max}(\pi) \le y$, в случае $k+1 \in N^2$ справедливо (3.47). Кроме того, из (3.38) следует(3.39) для всех $\pi \in \Pi(N_{k+1}, i)$, если $k+1 \in N^1$. Поскольку $(k+1) \in N^1$ либо $k+1 \in N^2$, то (3.34) при $m = k+1$ доказано.

Пусть теперь $\pi_{k+1}^i = \pi^{\emptyset}$. Тогда $\Pi_i = \emptyset$. Это означает, что $L_{\max}(\pi_i') > y$, $L_{\max}(\pi_i'') > y$. Из (3.38), (3.48) следует, что для любого расписания $\pi \in \Pi(N_{k+1}, i)$ справедливо (3.39), и неравенство (3.35) доказано.

Отметим, что $\pi_h = \pi_n^{\bar{t}} \in \Pi(N, \bar{t})$. Поскольку $\bar{t} = \max\{r_n, t\}$, то $\Pi(N, \bar{t}) = \Pi(N, t)$. Тогда из (3.34), (3.35) при $m = n$, $i = \bar{t}$ следует утверждение теоремы. Теорема доказана. □

Теорема 3.7 *Трудоёмкость процедуры h (алгоритма 3.9) составляет $O(n \log n + nP)$ операций.*

Доказательство. Для перенумерации требований согласно (3.11) достаточно $O(n \log n)$ операций [11]. На каждой итерации количество расписаний π_{k+1}^i не превышает P. На k-й итерации значения $L_{\max}(\pi_i')$, $L_{\max}(\pi_i'')$, $C_{\max}(\pi_i')$, $C_{\max}(\pi_i'')$ можно вычислять как

$$C_{\max}(\pi_i') = C_{\max}(\pi_k^{\max\{r_{k+1}, i\}+p_{k+1}}),$$

$$L_{\max}(\pi_i') = \max\{\max\{i, r_{k+1}\} + p_{k+1} - d_{k+1}, L_{\max}(\pi_k^{\max\{r_{k+1}, i\}+p_{k+1}})\},$$

$$C_{\max}(\pi_i'') = C_{\max}(\pi_k^i) + p_{k+1}, \tag{3.53}$$

$$L_{\max}(\pi_i'') = \max\{L_{\max}(\pi_k^i), C_{\max}(\pi_i'') - d_{k+1}\},$$

используя при этом значения $L_{\max}(\pi_k^{\max\{r_{k+1}, i\}+p_{k+1}})$, $L_{\max}(\pi_k^i)$, $C_{\max}(\pi_k^i)$, полученные на $(k-1)$-й итерации. Отметим, что согласно (3.11) $r_{k+1} \le r_j$ $\forall$ $j \in N_k$, т.е. между обслуживанием требований множества N_k и $k+1$ нет простоя прибора. Поэтому выполняется (3.53). Таким образом, на каждой итерации необходимо хранить не более P расписаний и соответствующих им значений целевой функции и моментов завершения. Напомним, что здесь мы рассматриваем случай, когда $p_j \in Z^+$ для всех $j \in N$. Значит трудоёмкость одной итерации – $O(P)$ операций, а поскольку процедура h состоит из n итераций, то её трудоёмкость составит $O(n \log n + nP)$ операций. Теорема доказана. □

Заметим, что применением процедуры h можно получить решение известной NP-полной задачи РАЗБИЕНИЕ ([7], с. 81) либо установить, что решения не существует. В этом параграфе предлагается способ решения задачи РАЗБИЕНИЕ, использующий процедуру построения допустимого относительно заданного значения y расписания для задачи минимизации максимального временно́го смещения при ограничениях (1.63) ($r_i \le r_j \;\Rightarrow\; d_i \ge d_j \;\forall\; i,j \in N$).

Сформулируем задачу РАЗБИЕНИЕ. Дано множество $N^0 = \{1, 2, ..., n^0\}$, каждому элементу i которого сопоставлено натуральное число a_j и выполняется $\sum\limits_{j \in N^0} a_j = 2A$, где A – натуральное число. Задача заключается в распознавании существует ли разбиение множества N^0 на два подмножества N_1^0 и N_2^0 такие, что

$$\sum_{j \in N_1^0} a_j = \sum_{j \in N_2^0} a_j = A. \tag{3.54}$$

В работе ([7], с. 293) построено сведение задачи о РАЗБИЕНИИ к задаче распознавания, соответствующей задаче $1|r_i \le r_j, d_i \ge d_j|L_{\max}$. Задача распознавания заключается в следующем. Существует ли расписание $\pi \in \Pi(N,t)$ такое, что $L_{\max}(\pi) \le y$ для заданного числа y. Сведение строится следующим образом. Полагаем $n = n^0 + 1$, $N = \{1, ..., n^0, n\}$; $p_j = a_j$, $r_j = 0$, $d_j = 2A + 1$, $j = \overline{1, n^0}$; $p_n = 1$, $r_n = A$, $d_n = A + 1$; $y = 0$. Доказано, что расписание $\pi = (\pi^1, n, \pi^2) \in \Pi(N,t)$, где $\pi^1 \in \Pi(N^1, t)$, $\pi^2 \in \Pi(N^2, C_{\max}(\pi^1, n))$, $N^1 \ne \emptyset$, $N^2 \ne \emptyset$, $N^1 \cup N^2 = N^0$, допустимо относительно заданного y тогда и только тогда, когда существует такое разбиение множества N^0 на подмножества $N_1^0 = N^1$ и $N_2^0 = N^2$, что справедливо (3.54). Необходимо заметить, что параметры требований множества N удовлетворяют ограничениям (1.63) ($r_i \le r_j \;\Rightarrow\; d_i \ge d_j \;\forall\; i,j \in N$), а допустимое расписание относительно заданного $y = 0$ в случае ограничений (1.63) можно построить процедурой h (алгоритмом 3.9), описанной в разделе 3.5.2. за $O(n \log n + nP)$ операций. Нетрудно видеть, что если $L_{\max}(\pi_h) \le 0$, то согласно определению параметров требований $N^1 \ne \emptyset$, $N^2 \ne \emptyset$. Таким образом, процедурой h можно получить решение задачи РАЗБИЕНИЕ либо установить, что решения не существует, за $O(n \log + nP)$ операций.

3.5.3 Алгоритм решения задачи $1|r_i \le r_j, d_i \ge d_j|L_{\max}$

В этом параграфе предлагается алгоритм решения задачи минимизации максимального временно́го смещения для случая (1.63) ($r_i \le r_j \;\Rightarrow\; d_i \ge$

$d_j \quad \forall \quad i,j \in N$), в котором используются процедуры h_0 (алгоритм 3.8) и h (алгоритм 3.9) описанные в разделах 3.5.1 и 3.5.2.

Идея алгоритма заключается в следующем. В качестве начального значения y полагается значение целевой функции расписания, построенного процедурой h_0. Далее, уменьшая значение y на единицу, не более, чем за $p_{\max}(N)$ вызова процедуры h, получаем оптимальное расписание.

Алгоритм 3.10 решения задачи $1 \mid r_i \le r_j, d_i \ge d_j \mid L_{\max}$

1: **Вспомогательный шаг.** Процедурой h_0 (алгоритм 3.8) строится расписание $\pi_{h_0} \in \Pi(N,t)$, и полагается $y^0 = L_{\max}(\pi_{h_0})$. Процедурой h (алгоритм 3.9), где $y = y^0$, строится допустимое относительно y^0 расписание $\pi_h^0 \in \Pi(N,t)$.
2: Пусть для $k \ge 1$ построено расписание π_h^{k-1}.
3: Построим π_h^k следующим образом. Полагаем $y^k = y^{k-1} - 1$ и процедурой h, где $y = y^k$, строится допустимое относительно y^k расписание $\pi_h^k \in \Pi(N,t)$.
4: **if** $\pi_h^k = \emptyset$ **then**
5: $\quad \pi^* = \pi_h^{k-1}$, и алгоритм заканчивает работу.
6: **end if**

Теорема 3.8 *Пусть на параметры требований множества N накладываются ограничения* (1.63). *Тогда алгоритмом* 3.10 *будет построено оптимальное расписание за $O(n^2P + np_{\max}(N)P)$ операций.*

Доказательство. Докажем, что $\pi_h^0 \neq \emptyset$. Допустим противное, т.е. $\pi_h^0 = \emptyset$. Тогда по теореме 3.6 $L_{\max}(\pi) > y^0 \ \forall \ \pi \in \Pi(N,t)$. Так как $\pi_{h_0} \in \Pi(N,t)$, то $L_{\max}(\pi_{h_0}) > y^0$, что противоречит выбору y^0. Таким образом, при указанном способе задания y^0 расписание π^*, построенное алгоритмом, таково, что $\pi^* \in \Pi(N,t)$.

Пусть $\pi_h^m = \emptyset$. Тогда по теореме 3.6 $L_{\max}(\pi) > y^m \ \forall \ \pi \in \Pi(N,t)$, а значит, $L_{\max}(\tilde{\pi}) > y^m \ \forall \ \tilde{\pi} \in \Pi^*(N,t)$. Очевидно, что y^m и $L_{\max}(\tilde{\pi})$ — целые числа для любого $\tilde{\pi} \in \Pi^*(N,t)$. Поэтому

$$L_{\max}(\tilde{\pi}) \ge y^{m-1} \ \ \forall \ \ \tilde{\pi} \in \Pi^*(N,t). \tag{3.55}$$

Так как согласно алгоритму 3.10 $\pi_h^{m-1} \neq \emptyset$, то по теореме 3.6

$$L_{\max}(\pi_h^{m-1}) \le y^{m-1}. \tag{3.56}$$

Поскольку $L_{\max}(\pi_h^{m-1}) \ge L_{\max}(\tilde{\pi}) \ \forall \ \tilde{\pi} \in \Pi^*(N,t)$, то из (3.55), (3.56) следует, что $\pi_h^{m-1} \in \Pi^*(N,t)$.

Оценим сложность алгоритма 3.10. Для выполнения начального шага потребуется $O(n^2P)$ операций, так как сложность процедур h_0 и h, соответственно, $O(n^2P)$ и $O(n\log n + nP)$ операций. На каждой итерации выполняется процедура h, поэтому каждая итерация имеет трудоёмкость $O(n\log n + nP)$

операций. Поскольку в качестве начального значения y^0 берётся значение целевой функции расписания, построенного процедурой h_0, то согласно теореме 3.4 значение целевой функции расписания π_{h_0} не более, чем на $p_{\max}(N)$, превосходит значение целевой функции оптимального расписания. Следовательно, количество итераций алгоритма 3.10 не превосходит $p_{\max}(N)$. Таким образом, трудоёмкость алгоритма составляет $O(n^2P+np_{\max}(N)P)$ операций. Теорема доказана. □

Отметим, что для случая (1.63) ($r_i \le r_j \;\Rightarrow\; d_i \ge d_j \;\;\forall\;\; i,j \in N$) алгоритм 3.10 даёт также способ построения N^* (алгоритм 1.7), трудоёмкость которой совпадает с трудоёмкостью алгоритма 3.10.

Таким образом для NP—трудной в сильном смысле задачи $1|r_i|L_{\max}$ построен псевдополиномиальный алгоритм трудоемкости $O(n^2P + np_{\max}P)$ операций.

3.5.4 Полиномиальные алгоритмы решения частных случаев задачи $1|r_i \le r_j, d_i \ge d_j|L_{\max}$

В этом параграфе предлагаются алгоритмы построения оптимальных расписаний задачи минимизации максимального временно́го смещения при ограничениях (1.63), (1.64) и (1.63), (1.65). Алгоритмы построены на основании свойств оптимальных расписаний, описанных выше.

Опишем алгоритм построения оптимального расписания задачи минимизации максимального временно́го смещения при ограничениях (1.63), (1.64):

$$r_i \le r_j \;\Rightarrow\; d_i \ge d_j \;\;\forall\;\; i,j \in N;$$

$$d_j - p_j \le d_{\min}(N) \;\;\forall\;\; j \in N.$$

Алгоритм 3.11 решения задачи при ограничениях: $r_i \le r_j \;\Rightarrow\; d_i \ge d_j \;\;\forall\;\; i,j \in N$; и $d_j - p_j \le d_{\min}(N) \;\;\forall\;\; j \in N$.

1: Полагаем $\pi^* = \arg\min\limits_{i=\overline{1,n}}\{L_{\max}(\pi_i)\}$, где $\pi_i = (\vec{\pi}_r, i)$, $\vec{\pi}_r \in \vec{\Pi}_r(N \setminus \{i\}, t)$, $i = 1, 2, ..., n$.

Теорема 3.9 *Пусть параметры требований множества N удовлетворяют ограничениям* (1.63), (1.64). *Тогда алгоритмом* 3.11 *будет построено оптимальное расписание $\pi^* \in \Pi(N,t)$ за $O(n^2)$ операций.*

Доказательство. Согласно лемме 1.13 найдётся такое $i \in N$, что расписание $\pi^* = (\vec{\pi}_r, i)$, где $\vec{\pi}_r \in \vec{\Pi}_r(N \setminus \{i\}, t)$, будет оптимальным, причём

$L_{\max}(\pi^*) = L_i(\pi^*)$. Поскольку на последнем месте может оказаться любое из n требований, то для нахождения оптимального расписания достаточно среди n расписаний π_i, $i = 1, 2, ..., n$, выбрать расписание с наименьшим значением целевой функции.

Оценим сложность алгоритма 3.11. Если первоначально перенумеровать требования по неубыванию моментов поступления, то для построения каждого расписания π_i и вычисления каждого значения $L_{\max}(\pi_i)$, $i = 1, ..., n$, потребуется $O(n)$ операций, а для построения всех расписаний π_i и вычисления всех значений $L_{\max}(\pi_i)$ необходимо $O(n^2)$ операций. Для выбора π^* из n расписаний достаточно $O(n)$ операций. Поскольку для перенумерации необходимо $O(n \log n)$ операций [11], то трудоёмкость алгоритма 3.11 составляет $O(n^2)$ операций. Теорема доказана. □

Отметим, что для случая (1.63), (1.64) алгоритм 3.11 даёт также способ построения N^* (алгоритм 1.7), трудоёмкость которой в этом случае совпадает с трудоёмкостью алгоритма 3.11.

Теорема 3.10 *Пусть параметры требований множества N удовлетворяют ограничениям* (1.63), (1.65)*: $r_i \le r_j \Rightarrow d_i \ge d_j \quad \forall \quad i, j \in N$ и $r_i \le r_j \Rightarrow d_i - p_i \ge d_j \quad \forall \quad i, j \in N, \quad i \ne j$. Тогда оптимальное расписание $\pi^* \in \vec{\Pi}_d(N, t)$ строится за $O(n \log n)$ операций.*

Доказательство. Согласно лемме 1.14 любое расписание $\pi \in \vec{\Pi}_d(N, t)$ будет оптимальным. Поэтому для построения оптимального расписания π^* достаточно перенумеровать требования множества N по неубыванию директивных сроков и положить $\pi^* = (1, ..., n)$. Поскольку для перенумерации требований достаточно $O(n \log n)$ операций, то трудоёмкость построения оптимального расписания составит $O(n \log n)$ операций. □

Из теоремы 3.10 следует, что для построения оптимального расписания для случая (1.63), (1.65) достаточно упорядочить требования по неубыванию директивных сроков. Отметим, что для случая (1.63), (1.65) в алгоритме 1.7 $N^* = \emptyset$, и трудоёмкость алгоритма 1.7 совпадает с трудоёмкостью перенумерации требований.

Предоставим возможность заинтересованному читателю выделить другие полиномиально разрешимые случаи исследуемой задачи.

3.6 Приближённый алгоритм решения общего случая задачи. Оценка абсолютной погрешности

В этом разделе предлагается приближённый алгоритм решения задачи минимизации максимального временно́го смещения в общем случае, использующий алгоритм решения частного случая задачи при ограничениях (1.63). Идея алгоритма заключается в следующем. Директивные сроки требований изменяются так, чтобы новые параметры требований удовлетворяли ограничениям (1.63). Далее задача решается алгоритмом 3.10. Полученное расписание является приближённым расписанием с минимальным значением границы абсолютной погрешности оптимального значения целевой функции ($L_{\max}$) на множестве оптимальных расписаний частного случая при условиях (1.63). Минимизация границы абсолютной погрешности обеспечивается алгоритмом изменения директивных сроков.

Не ограничивая общности, будем полагать, что требования множества N пронумерованы так, что

$$r_1 \leq r_2 \leq ... \leq r_n, \tag{3.57}$$

причём, если

$$r_j = r_{j+1} \ \Rightarrow d_j \geq d_{j+1} \ \ \forall \ \ j = 1, ..., n-1. \tag{3.58}$$

Сформулируем и решим следующую задачу:

$$\begin{cases} \max\limits_{j \in N}\{d_j - d'_j\} - \min\limits_{j \in N}\{d_j - d'_j\} \longrightarrow \min\limits_{d'_1, d'_2, ..., d'_n} \\ d'_1 \geq d'_2 \geq ... \geq d'_n. \end{cases} \tag{3.59}$$

Решение задачи (3.59) может быть найдено следующим алгоритмом.

Алгоритм 3.12 решения задачи (3.59)

1: **Вспомогательный шаг** Перенумеруем требования множества N согласно (3.57), (3.58). Полагаем $N_1 = N$.
2: **Основной шаг** Пусть для $k \geq 1$ построено множество N_k.
3: **if** $N_k \neq \emptyset$ **then**
4: $l_k = \max\{j \in N_k : d_j = \max\limits_{i \in N_k} d_i\}$, $\bar{N}_k = \{j \in N_k : j \leq l_k\}$, и полагаем $\bar{d}'_j = d_{l_k} \ \forall \ j \in \bar{N}_k$, $N_{k+1} = N_k \setminus \bar{N}_k, k = k+1$ и переходим к следующему шагу.
5: **else**
6: алгоритм заканчивает работу.
7: **end if**

Лемма 3.3 *Решение задачи* (3.59) *может быть найдено алгоритмом* 3.12 *за* $O(n \log n)$ *операций.*

Доказательство. Оценим трудоёмкость алгоритма 3.12. Для перенумерации требований понадобится $O(n \log n)$ операций. Поскольку $\bar{N}_k \cap \bar{N}_{k+1} = \emptyset$, то для нахождения всех $\bar{d}'_j \ \forall \ j \in N$ потребуется не более $O(n)$ операций. Поэтому, трудоёмкость алгоритма 3.12 составит $O(n \log n)$ операций.

Пусть число итераций алгоритма 3.12 равно m. Согласно алгоритму 3.12 значения $\bar{d}'_j \ \forall \ j = 1, ..., n$ удовлетворяют ограничениям

$$\bar{d}'_1 \geq \ldots \geq \bar{d}'_n \tag{3.60}$$

и, кроме того,

$$\bar{d}'_j \geq d_j, \ j = 1, 2, \ldots, n. \tag{3.61}$$

Покажем, что

$$\max_{j \in N}\{\bar{d}'_j - d_j\} - \min_{j \in N}\{\bar{d}'_j - d_j\} \leq \max_{j \in N}\{d'_j - d_j\} - \min_{j \in N}\{d'_j - d_j\} \tag{3.62}$$

для любых целых чисел d'_j, удовлетворяющих (3.59). Тогда утверждение леммы будет доказано. Для этого выберем произвольные целые значения $\tilde{d}'_j$, $j = 1, \ldots, n$, такие, что

$$\tilde{d}'_1 \geq \ldots \geq \tilde{d}'_n, \tag{3.63}$$

и проверим справедливость неравенства (3.62) при $d'_j = \tilde{d}'_j \ \forall \ j = 1, \ldots, n$.

Очевидно, что $\max\{d_j - d'_j\} = -\min\{d'_j - d_j\}$ и $\min\{d_j - d'_j\} = -\max\{d'_j - d_j\} \ \forall \ d'_1, \ldots, d'_n$. Отсюда

$$\max_{j \in N}\{d_j - \bar{d}'_j\} - \min_{j \in N}\{d_j - \bar{d}'_j\} = \max_{j \in N}\{\bar{d}'_j - d_j\} - \min_{j \in N}\{\bar{d}'_j - d_j\}, \tag{3.64}$$

и

$$\max_{j \in N}\{d_j - \tilde{d}'_j\} - \min_{j \in N}\{d_j - \tilde{d}'_j\} = \max_{j \in N}\{\tilde{d}'_j - d_j\} - \min_{j \in N}\{\tilde{d}'_j - d_j\}. \tag{3.65}$$

Поскольку требования множества N пронумерованы согласно (3.57), (3.58), то требования $l_1, \ldots, l_k$ и множества $\bar{N}_1, \ldots, \bar{N}_k$ можно выбрать по алгоритму 3.12. Очевидно, что $\bar{N}_k \cap \bar{N}_{k+1} = \emptyset$ и $\bar{N}_1 \cup \ldots \cup \bar{N}_m = N$. Отсюда

$$\begin{gathered} \max_{j \in N}\{\bar{d}'_j - d_j\} - \min_{j \in N}\{\bar{d}'_j - d_j\}) = \\ \max_{1 \leq k \leq m}(\max_{j \in \bar{N}_k}\{\bar{d}'_j - d_j\}) - \min_{1 \leq k \leq m}(\min_{j \in \bar{N}_k}\{\bar{d}'_j - d_j\}) \end{gathered} \tag{3.66}$$

и

$$\begin{gathered} \max_{j \in N}\{\tilde{d}'_j - d_j\} - \min_{j \in N}\{\tilde{d}'_j - d_j\}) = \\ \max_{1 \leq k \leq m}(\max_{j \in \bar{N}_k}\{\tilde{d}'_j - d_j\}) - \min_{1 \leq k \leq m}(\min_{j \in \bar{N}_k}\{\tilde{d}'_j - d_j\}). \end{gathered} \tag{3.67}$$

Обозначим через $s_k \in \bar{N}_k$ такое требование, что

$$d_{s_k} \le d_j \ \ \forall \ \ j \in \bar{N}_k, \ \ \ k = 1, ..., m. \tag{3.68}$$

По алгоритму 3.12

$$\bar{d}'_j = \bar{d}'_{l_k} = d_{l_k} \ \ \forall \ \ j \in \bar{N}_k, \ \ \ k = 1, ..., m. \tag{3.69}$$

Согласно выбору требования l_k с учётом (3.68), (3.69)

$$\max_{j \in \bar{N}_k} \{\bar{d}'_j - d_j\} = d_{l_k} - d_{s_k} \ \ \forall \ \ k = 1, ..., m, \tag{3.70}$$

$$\min_{j \in \bar{N}_k} \{\bar{d}'_j - d_j\} = 0 \ \ \forall \ \ k = 1, ..., m, \tag{3.71}$$

и $d_{l_k} \ge d_j$ для всех $j \in \bar{N}_k$. Кроме того, в силу (3.63) $\tilde{d}'_{l_k} \le \tilde{d}'_j$ для всех $j \in \bar{N}_k$, $k = 1, ..., m$. Отсюда

$$\min_{j \in \bar{N}_k} \{\tilde{d}'_j - d_j\} = \tilde{d}'_{l_k} - d_{l_k} \ \ \forall \ \ k = 1, ..., m. \tag{3.72}$$

Пусть числа C_j, $j = 1, ..., n$, таковы, что $\tilde{d}'_j = \bar{d}'_j + C_j$. Тогда с учётом (3.69)

$$\tilde{d}'_j = d_{l_k} + C_j \ \ \forall \ \ j \in \bar{N}_k, \ \ \ k = 1, ..., m, \tag{3.73}$$

а из (3.63), (3.73) следует, что

$$i \le j \ \Rightarrow \ C_i \ge C_j \ \ \forall \ \ i, j \in \bar{N}_k, \ \ \ k = 1, ..., m. \tag{3.74}$$

Так как $\max\limits_{j \in \bar{N}_k} \{\tilde{d}'_j - d_j\} \ge \tilde{d}'_j - d_j \ \forall \ j \in \bar{N}_k$, то $\max\limits_{j \in \bar{N}_k} \{\tilde{d}'_j - d_j\} \ge \tilde{d}'_{s_k} - d_{s_k}$. Отсюда с учётом (3.73)

$$\max_{j \in \bar{N}_k} \{\tilde{d}'_j - d_j\} \ge d_{l_k} + C_{s_k} - d_{s_k} \ \ \forall \ \ k = 1, ..., m. \tag{3.75}$$

Из (3.70), (3.71), (3.72), (3.75) следует, что $\max\limits_{j \in \bar{N}_k} \{\bar{d}'_j - d_j\} - \min\limits_{j \in \bar{N}_k} \{\bar{d}'_j - d_j\} - (\max\limits_{j \in \bar{N}_k} \{\tilde{d}'_j - d_j\} - \min\limits_{j \in \bar{N}_k} \{\tilde{d}'_j - d_j\}) \le \tilde{d}'_{l_k} - d_{l_k} - C_{s_k} \ \forall \ k = 1, ..., m$. Согласно выбору требований l_k и s_k имеем, что $s_k \le l_k$. Тогда из (3.73) с учётом (3.74) следует, что $\tilde{d}'_{l_k} \le d_{l_k} + C_{s_k}$ для всех $k = 1, ..., m$. Отсюда

$$\max_{j \in \bar{N}_k} \{\bar{d}'_j - d_j\} - \min_{j \in \bar{N}_k} \{\bar{d}'_j - d_j\} \le \max_{j \in \bar{N}_k} \{\tilde{d}'_j - d_j\} - \min_{j \in \bar{N}_k} \{\tilde{d}'_j - d_j\} \ \forall \ k = 1, ..., m,$$

и в силу (3.66), (3.67)

$$\max_{j \in N} \{\bar{d}'_j - d_j\} - \min_{j \in N} \{\bar{d}'_j - d_j\} \le \max_{j \in N} \{\tilde{d}'_j - d_j\} - \min_{j \in N} \{\tilde{d}'_j - d_j\}.$$

Отсюда с учётом (3.64), (3.65) следует неравенство (3.62) при $d'_j = \tilde{d}'_j$. Лемма доказана. □

Опишем приближённый алгоритм решения задачи минимизации максимального временно́го смещения в общем случае.

Алгоритм 3.13 решения задачи (3.59)

1: Для требований множества N алгоритмом 3.12 находим новые директивные сроки $\bar{d}'_j$, $j \in N$.
2: Алгоритмом 3.10 решаем задачу $1|r_i \le r_j, d_i \ge d_j|L_{\max}$ при $d_j = \bar{d}'_j$.

Теорема 3.11 *Алгоритмом* 3.13 *трудоёмкости* $O(n^2P + np_{\max}P)$ *операций строится расписание* $\pi' \in \Pi(N,t)$ *такое, что* $L_{\max}(\pi') - L_{\max}(\pi^*) \le \bar{\rho}_d$, *где* $\pi^* \in \Pi^*(N,t)$, $\bar{\rho} = \max\limits_{j\in N}\{d_j - \bar{d}'_j\} - \min\limits_{j\in N}\{d_j - \bar{d}'_j\}$, *и* $\bar{\rho}_d$ – *минимальное значение разности* $\max\limits_{j\in N}\{d_j - d'_j\} - \min\limits_{j\in N}\{d_j - d'_j\}$ *по всем* $d'_1, ..., d'_n$, *удовлетворяющих неравенствам* (3.59).

Доказательство. Из леммы 1.16 имеем, что $L_{\max}(\pi') \le L_{\max}(\pi^*) + \rho_d$, где $\pi^*, \pi' \in \Pi(N,t)$ — оптимальные расписания обслуживания требований множества N с момента времени t при директивных сроках d_j и d'_j соответственно. Из леммы 3.3 следует, что минимальное значение ρ_d достигается при $d'_j = \bar{d}'_j$, $j = \overline{1,n}$, найденное алгоритмом 3.12.

Трудоёмкость алгоритма 3.12 не превышает $O(n \log n)$ операций. А трудоёмкость алгоритма 3.10 – $O(n^2P + np_{\max}P)$ операций, поэтому трудоёмкость алгоритма 3.13 составляет $O(n^2P + np_{\max}P)$ операций. Теорема доказана.

□

Таким образом, изменив директивные сроки исходной задачи и решая полученную задачу алгоритмом для частного случая (1.63) за $O(n^2P + np_{\max}P)$ операций получим приближённое решение исходной задачи с минимальной величиной абсолютной погрешности оптимального значения целевой функции ($L_{\max}$) на множестве оптимальных расписаний задачи при условиях (1.63) ($r_i \le r_j \;\Rightarrow\; d_i \ge d_j \;\; \forall \;\; i,j \in N$).

Как было показано в главе 1, можно изменять не только директивные сроки, но и моменты поступления и продолжительности обслуживания требований.

Для случая (1.63) верно $r_1 \le r_2 \le \ldots \le r_n$ и $d_1 \ge d_2 \ge \ldots \ge d_n$. Задача линейного программирования (1.88) нахождения для произвольного примера $A = \{(r_j^A, p_j^A, d_j^A) \mid j \in N\}$ примера $B = \{(r_j^B, p_j^B, d_j^B) \mid j \in N\}$, принадлежащего (1.63), на котором достигается минимальная оценка абсолютной погрешности, формулируется следующим образом.

$$\begin{cases} (x^d - y^d + x^r - y^r) + \sum\limits_{j \in N} x_j^p \longrightarrow \min \\ y^d \le d_j^A - d_j^B \le x^d, \quad \forall j \in N, \\ y^r \le r_j^A - r_j^B \le x^r, \quad \forall j \in N, \\ -x_j^p \le p_j^A - p_j^B \le x_j^p, \quad \forall j \in N, \\ 0 \le x_j^p, \quad \forall j \in N, \\ CR^B \le 0, \\ -CD^B \le 0, \end{cases}$$

где матрица C размерности $(n-1) \times n$ имеет вид

$$C = \begin{pmatrix} 1 & -1 & 0 & 0 \ldots & 0 & 0 \\ 0 & 1 & -1 & 0 \ldots & 0 & 0 \\ 0 & 0 & 1 & -1 \ldots & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & 0 \ldots & -1 & 0 \\ 0 & 0 & 0 & 0 \ldots & 1 & -1 \end{pmatrix}.$$

Решив данную задачу линейного программирования, мы найдем пример B, принадлежащий (1.63), наиболее близкий к исследуемому примеру A во введённой нами метрике.

Глава 4

Алгоритм ветвей и отсечений для решения задачи $1 \mid r_j \mid \sum w_j U_j$

В данной главе мы рассмотрим алгоритм ветвей и отсечений для решения задачи минимизации взвешенного числа запаздывающих требований на одном приборе. Алгоритм разработан в рамках "гибридной" схемы, представленной для полноты изложения в разделе 4.1. В разделе 4.2 будет описана постановка задачи, а также сделан обзор литературы по существующим методам её решения. В разделе 4.3 задача будет сформулирована как задача частично целочисленного линейного программирования и затем переформулирована как задача вида (GP – гибридная формулировка оптимизационной задачи). В разделе 4.3.1 мы представим дополнительные ограничения, цель которых улучшить релаксацию (MIP) задачи (GP). Далее в разделе 4.4 будут рассмотрены алгоритмы построения отсечений, основанные на методах решения задачи $1 \mid r_j \mid L_{\max}$, представленных в главе 3 настоящей работы. Наконец, в разделах 4.5 и 4.6 будет представлен "гибридный" алгоритм ветвей и отсечений и проведено его экспериментальное сравнение с другими лучшими на сегодняшний день алгоритмами решения задачи $1 \mid r_j \mid \sum w_j U_j$.

4.1 "Гибридная" схема решения одного класса задач целочисленного линейного программирования

В этом разделе мы представим схему решения класса задач целочисленного программирования. В следующих разделах показывается, что некоторые задачи теории расписаний могут быть сформулированы как задачи данного класса и решены "гибридным" алгоритмом в рамках представленной схемы.

"Гибридные" алгоритмы базируются на декомпозиции исходной задачи, когда пространство переменных разбивается на два подпространства. Дан-

ное разбиение является обобщением декомпозиции Бендерса (Benders [88]). После декомпозиции основная задача содержит переменные одного подпространства, а подзадача (или подзадачи) – другого. Отличительной чертой "гибридных" алгоритмов является то, что задача и подзадача решаются разными методами. Такими методами могут быть [частично] целочисленное линейное программирование, линейное программирование, программирование в ограничениях или специализированные комбинаторные алгоритмы. Основным преимуществом такого подхода как раз и является кооперация (гибридизация) различных по своей природе методов. Существуют задачи, которые не могут быть успешно решены ни одним из методов отдельно. Для таких задач и представляется перспективным разработка "гибридных" схем (методов) решения и основанных на них алгоритмов.

В русскоязычной литературе алгоритмы ветвей и отсечений, а также "гибридные" алгоритмы рассматривались в работах Корбута, Сигала и Финкельштейна [15, 67]. В последние время в англоязычной литературе появилось множество работ на данную тему. В данном разделе мы будем придерживаться в основном статье Джейна и Гроссмана (Jain, Grossmann [134]), так как предложенная ими схема решения наилучшим образом подходит для рассматриваемой нами задачи. В конце раздела будет сделан обзор некоторых других работ данного направления.

Итак, пусть рассматриваемая задача может быть сформулирована как следующая задача частично целочисленного линейного программирования (*ЧЦЛП*):

$$(\mathrm{OP}) : c^T x \longrightarrow \min \tag{4.1}$$

$$Ax + By + Cv \leq a; \tag{4.2}$$

$$A'x + B'y + C'v \leq a'; \tag{4.3}$$

$$x \in \{0,1\}^n,\ y \in \{0,1\}^m,\ v \in \mathbb{R}^p. \tag{4.4}$$

Задача (OP) имеет как целочисленные (булевы), так и действительные переменные, причём только часть булевых переменных имеет ненулевые коэффициенты в целевой функции. Множество ограничений разделено на два подмножества. Ограничения (4.2) отображают некоторые аспекты задачи, которые могут быть эффективно смоделированы линейными ограничениями. Напротив, ограничения (4.3) вносят сложность в решение задачи ЧЦЛП стандартными методами, например, ввиду большой размерности. Важным моментом, влияющим на эффективность рассматриваемой схемы, является то, что отсутствие ограничений (4.3) по возможности не должно сильно влиять на оптимальное значение линейной релаксации задачи (OP).

Теперь мы представим “гибридную” формулировку задачи (OP):

$$
\begin{aligned}
(\mathrm{GP}) : c^T x &\longrightarrow \min && (4.5)\\
Ax + By + Cv &\le a; && (4.6)\\
x &\Leftrightarrow \bar{x}; && (4.7)\\
\bar{G}(\bar{x}, \bar{y}, \bar{v}) &\le 0; && (4.8)\\
x \in \{0,1\}^n,\ y &\in \{0,1\}^m,\ v \in \mathbb{R}^p; && (4.9)\\
\bar{x}, \bar{y}, \bar{v} &\in \mathbb{D}. && (4.10)
\end{aligned}
$$

Здесь ограничения (4.8) и (4.10) – это переформулированные ограничения (4.3). Ограничения (4.8) могут быть любыми, они не обязательно должны быть линейными. Множество возможных значений $\mathbb{D}$ переменных $\bar{x}$, $\bar{y}$, $\bar{v}$ также может быть задано произвольным способом. Однако, между переменными x и $\bar{x}$ должно быть взаимно-однозначное соответствие (4.7).

Основой схемы решения задачи (GP) является задача ЧЦЛП, включающая только часть исходных ограничений и решаемая стандартным методом, а также задача выполнимости, которая решается наиболее подходящими для нее методами, это может быть программирование в ограничениях или специализированные комбинаторные алгоритмы. Решая задачу ЧЦЛП, мы находим решение, удовлетворяющее ограничениям (4.6) и (4.9), оптимизируя при этом целевую функцию (4.5). Решение x данной задачи преобразуется в частичное решение $\bar{x}$ задачи выполнимости. Затем задача выполнимости проверяет существуют ли такие значения $\bar{y}$ и $\bar{v}$ для переменных y и v, что решение $(\bar{x}, \bar{y}, \bar{v})$ является допустимым для ограничений (4.8) и (4.10). Если расширение возможно, то значение целевой функции такого полного решения такое же, как и частичного решения $\bar{x}$.

Опишем детально рассматриваемую схему решения задачи (GP). На итерации k алгоритма, $k = 1, 2, \ldots$, решается следующая задача ЧЦЛП:

$$
\begin{aligned}
(\mathrm{MIP}) : c^T x &\longrightarrow \min && (4.11)\\
Ax + By + Cv &\le a; && (4.12)\\
Q^k x &\le q^k \quad \forall k \in \{1, 2, \ldots, k-1\}; && (4.13)\\
x \in \{0,1\}^n,\ y &\in \{0,1\}^m,\ v \in \mathbb{R}^p. && (4.14)
\end{aligned}
$$

Задача (MIP) может быть решена стандартным методом ветвей и границ (метод Лэнд и Дойг [143]). Ограничения (4.13) находятся на предыдущих итерациях, они будут рассмотрены далее по тексту. Если задача (MIP) не имеет решения, то и исходная задача (GP) также не имеет решения, т.к. область допустимых решений (MIP) больше чем (GP). Если найдено решение

x^k задачи (MIP), то оно трансформируется в $\bar{x}^k$ с помощью соотношений (4.7) и решается следующая задача выполнимости:

$$(\text{FP}) : \bar{y}, \bar{v} \longrightarrow \text{find} \tag{4.15}$$

$$\bar{G}(\bar{x}^k, \bar{y}, \bar{v}) \leq 0; \tag{4.16}$$

$$\bar{y}, \bar{v} \in \mathbb{D}. \tag{4.17}$$

В некоторых случаях задача (FP) разбивается на независимые подзадачи, каждая из которых решается по отдельности, что уменьшает время решения. Поэтому, когда это возможно, следует разбивать ограничения задачи на подмножества (4.2) и (4.3) таким образом, чтобы задача (FP) разбивалась на подзадачи.

Если решение $(\bar{x}^k, \bar{y}, \bar{v})$ задачи (FP) найдено, то оно будет оптимальным для исходной задачи (GP), так как это решение имеет такое же значение целевой функции, как и решение x^k задачи (MIP). Если же решение не найдено, то x^k исключается из пространства допустимых решений задачи (MIP). Для этого генерируется следующее *отсечение*:

$$\sum_{i \in T^k} x_i - \sum_{i \in F^k} x_i \leq B^k - 1, \tag{4.18}$$

где $T^k = \{i \mid x_i^k = 1\}$, $F^k = \{i \mid x_i^k = 0\}$, $B^k = \mid T^k \mid$. Отсечения в форме (4.18) являются универсальными для рассматриваемого класса задач и отсекают только одно решение, так как любое другое решение $x' \neq x^k$ удовлетворяет ограничению (4.18). Поэтому, когда это возможно, следует использовать специфические для конкретной задачи более сильные отсечения $Q^k x \leq q^k$, которые отсекают несколько однотипных решений из области допустимых значений задачи (MIP). Отсечения являются связующим звеном между задачами (MIP) и (FP), и от того, какой тип отсечений используется, напрямую зависит эффективность всего алгоритма, разработанного в рамках рассматриваемой схемы.

Формально описанная схема решения задачи (GP) представлена как алгоритм 4.1.

Теорема 4.1 *[134] Алгоритм 4.1 решает задачу (GP) или доказывает её недопустимость за конечное число итераций.* □

Представленная схема имеет один существенный недостаток. На каждой итерации необходимо решить задачу ЧЦЛП. Это неэффективно при решении примеров большой размерности, так как для выполнения алгоритма

Алгоритм 4.1 "Гибридная" схема решения задачи (GP) итеративным методом

```
1: k := 0
2: repeat
3:    найдём решение x^k задачи (MIP)
4:    if задача (MIP) не имеет решения then
5:       задача (GP) не имеет решения; stop
6:    end if
7:    преобразуем x^k в x̄^k с помощью соотношений (4.7)
8:    решим задачу (FP)
9:    if задача (FP) не имеет решения then
10:      построим отсечение (4.18)
11:      добавим построенное отсечение в (MIP) как ограничение Q^k x ≤ q^k
12:      k := k + 1
13:   else
14:      (x̄^k, ȳ, v̄) – оптимальное решение задачи (GP); stop
15:   end if
16: until false
```

ветвей и границ, решающего задачу (MIP), может потребоваться долгое время, причём на каждой итерации.

Намного более эффективным способом является решение задачи ЧЦЛП всего один раз. При этом каждое целочисленное решение, полученное на каком-либо узле глобального дерева поиска метода ветвей и границ, проверяется на допустимость задачей (FP). В случае недопустимости решения отсечения динамически добавляются к исходным ограничениям задачи ЧЦЛП. Такой подход получил название метода *ветвей и проверок* (Branch-and-Check [213]).

Опишем схему метода ветвей и проверок более детально. На каждом узле глобального дерева поиска решается следующая задача линейного программирования.

$$
\begin{aligned}
&(\mathrm{LP}): c^T x \longrightarrow \min = z^* && (4.19)\\
&Ax + By + Cv \le a; && (4.20)\\
&Qx \le q; && (4.21)\\
&x_j^{LB} \le x \le x_j^{UB}, \quad \forall j \in \{1,\dots,n\}; && (4.22)\\
&y_i^{LB} \le y \le y_i^{UB}, \quad \forall i \in \{1,\dots,m\}; && (4.23)\\
&x \in \mathbb{R}^n,\ y \in \mathbb{R}^m,\ v \in \mathbb{R}^p; && (4.24)\\
&x^{LB}, x^{UB} \in \{0,1\}^n, \quad y^{LB}, y^{UB} \in \{0,1\}^m. && (4.25)
\end{aligned}
$$

Разница между задачами линейного программирования, решаемыми на различных узлах глобального дерева поиска, заключается в разных минимальных и максимальных значениях переменных x^{LB}, x^{UB}, y^{LB}, y^{UB}, а также

разном наборе отсечений (4.21). Если задача (LP) недопустима или значение её оптимального решения больше значения целевой функции текущего лучшего найденного решения, то текущий узел глобального дерева поиска отсекается. Иначе, если решение задачи (LP) не является целочисленным, то выбирается переменная x_j или y_i, которая в оптимальном решении принимает нецелочисленное значение, и создаются два дочерних узла, в одном из которых принимается $x_j^{UB} = 0$ или $y_i^{UB} = 0$, а в другом – $x_j^{LB} = 1$ или $y_i^{LB} = 1$. Если же решение задачи линейного программирования целочисленное, то оно проверяется на допустимость задачей (FP). В случае недопустимости соответствующее отсечение (4.18) добавляется к текущему множеству отсечений (4.21) и задача (LP) решается снова. Если решение признано допустимым, то лучшее текущее решение обновляется, а текущий узел обрезается. Схема решения рассматриваемого класса задач "гибридным" методом ветвей и проверок формально представлена как алгоритм 4.2.

Алгоритм 4.2 "Гибридная" схема решения задачи (GP) методом ветвей и проверок

создадим задачу p^0, где $x^{LB} = \mathbf{0}^n$, $x^{UB} = \mathbf{1}^n$, $y^{LB} = \mathbf{0}^m$, $y^{UB} = \mathbf{1}^m$, множество отсечений $Qx \le q$ пустое
$UB := +\infty$; $P := \{p^0\}$; $l := 0$
while $P \neq \emptyset$ **do**
  выберем задачу p вида (LP) из P; $P := P \setminus \{p\}$;
  решим задачу p
  if p имеет решение (x^*, y^*, v^*) **AND** $z^* < UB$ **then**
    if (x^*, y^*) – целочисленный **then**
      преобразуем x^* в $\bar{x}$ с помощью соотношений (4.7);
      решим задачу (FP)
      if задача (FP) не имеет решения **then**
        $l := l + 1$; построим отсечение $Q^l x \le q^l$ вида (4.18);
        $P := P \cup \{p\}$; добавим построенное отсечение во все задачи из P
      else
        $UB := z^*$; запомним решение $(\bar{x}, \bar{y}, \bar{v})$
      end if
    else
      выберем переменную $x_j :\ x_j^* \notin \{0, 1\}$ или $y_i :\ y_i^* \notin \{0, 1\}$;
      скопируем задачу p в p' и p'';
      положим $x_j^{UB} = 0$ или $y_i^{UB} = 0$ в задаче p'; $P := P \cup \{p'\}$;
      положим $x_j^{LB} = 1$ или $y_i^{LB} = 1$ в задаче p''; $P := P \cup \{p''\}$
    end if
  end if
end while

В последнее время в литературе появилось множество работ с различными подходами к декомпозиции задач и решения их частей разными методами. Из них можно выделить следующие результаты.

Бокмайр и Писарук (Bockmayr, Pisaruk [90]) выделили в рамках рассматриваемой в этом разделе схемы специальный случай, когда ограничения (4.8) являются *монотонными*. Ограничение $G(x_1, \ldots, x_n) = 0$ является монотонным, если монотонна функция $G: \ x \to \{0,1\}$, где $x \in \{0,1\}^n$ (функция G монотонна, если $\forall x, x' \in \{0,1\}^n$, $x \leq x'$, выполняется $G(x) \leq G(x')$). Если в алгоритме 4.1 или алгоритме 4.2 решение задачи (FP) показало, что x^k является недопустимым, то в случае, если ограничения (4.8) являются монотонными, можно построить более сильное, чем (4.18) отсечение:

$$\sum_{i \in T^k} x_j \leq B^k - 1, \tag{4.26}$$

где $T^k = \{i \mid x_i^k = 1\}$, $B^k = \mid T^k \mid$. В работе также показано, что если задача (GP) решается методом ветвей и проверок, то отсечения (4.26) могут быть построены не только для целочисленного решения, но и для дробных решений линейной релаксации (LP).

Торстейнсон (Thorsteinsson [213]) разработал схему метода ветвей и проверок (термин Branch-and-Check был также введен этим автором) для более общего класса задач, чем (GP), однако в его работе нет четкого описания алгоритма построения отсечений, автор предлагает разрабатывать собственный такой алгоритм для каждого конкретного класса задач. Также в этой работе отмечена важность добавления релаксации ограничений (4.8) в задачу (MIP), при этом релаксация не должна сильно затруднять решение задачи. Этот шаг позволяет сокращать число отсечений, требуемых для решения задач в рамках рассматриваемой схемы.

Хукер и Оттосон (Hooker, Ottosson [131]) обобщили метод декомпозиции Бендерса на широкий спектр оптимизационных задач. Ими были разработаны алгоритмы для построения отсечений для задачи пропозициональной выполнимости, а также для задачи булевого линейного программирования.

4.2 Постановка задачи

Рассмотрим постановку задачи минимизации взвешенного числа запаздывающих требований на одном приборе. Множество из n требований $N = \{1, 2, \ldots, n\}$ необходимо обслужить на одном приборе. Запрещается одновременное обслуживание и прерывания при обслуживании требований. Каждое требование $j \in N$ имеет время поступления r_j, время обслуживания p_j, директивный срок d_j и вес w_j. Требование $j \in N$ обслуживается *вовремя* при расписании π, если момент окончания обслуживания данного

требования не превышает его директивный срок: $C_j(\pi) \leq d_j$, иначе требование j *запаздывает*. В стандартной нотации признак запаздывания требования j обозначается как U_j: $U_j = 1$ если j запаздывает, и $U_j = 0$ если j обслуживается вовремя. Для решения задачи требуется найти расписание, при котором минимизируется взвешенное число запаздывающих требований. В стандартной нотации теории расписаний [125] данная задача обозначается как $1 \mid r_j \mid \sum w_j U_j$. Задача минимизации взвешенного числа запаздывающих требований на одном приборе является одной из классических задач теории расписаний.

Задача является NP-трудной в сильном смысле даже при отсутствии весов [167]. Однако существует несколько полиномиально разрешимых случаев задачи. Мур (Moore [176]) показал, что случай $1 \mid\mid \sum U_j$ без весов и времён поступлений разрешим за $O(n^2)$ операций.

Мы внесли некоторые модификации в алгоритм Мура, — алгоритм 4.3, трудоемкость которого составляет $O(n \log n)$ операций. Так как изменения в алгоритме не носят принципиального характера, то мы оставим за ним название алгоритма Мура.

Полиномиальность случая без весов, когда времена поступления и директивные сроки упорядочены одинаково ($r_i < r_j \Rightarrow d_i \leq d_j$), была доказана Кизом и др. (Kise et al. [139]), $O(n \log n)$ операций алгоритм для этого случая был предложен Лоулером (Lawler [146]). Также Лаулером (Lawler [145]) был разработан полиномиальный алгоритм динамического программирования решающий задачу за время $O(n^5)$ в случае, когда отсутствуют веса, но разрешены прерывания обслуживания ($1 \mid r_j; pmtn \mid \sum U_j$). В дальнейшем этот алгоритм был улучшен до сложности $O(n^4)$ Баптистом (Baptiste [81]).

Алгоритм 4.3 Мура решения задачи $1 \mid\mid \sum U_j$

1: создадим упорядоченный список множества N:
   $N_1 = \{i_1, i_2, \ldots, i_n\}$;
   где $d_{i_k} \leq d_{i_{k+1}}$, $k = 1, 2, \ldots, n-1$;
   $N_e := \emptyset$; $j_0 := i_1$;
2: $N_0 := \emptyset$; $t = 0$; $k := 1$; $\pi = \emptyset$;
3: while $k \leq n$ do
4: $t := t + p_{i_k}$
5: if $p_{j_0} < p_{i_k}$ then $j_0 := i_k$;
6: if $t \leq d_{i_k}$ then $\pi := (\pi, i_k)$
   else $N_e := N_e \cup \{j_0\}$; $\pi := \pi \backslash \{j_0\}$; $t := t - p_{j_0}$;
7: $k := k + 1$;
8: end while

Для NP-сложного в обычном смысле специального случая задачи с

весами, но без времён поступления $1 \mid\mid \sum w_j U_j$ Лоулером и Муром (Lawler, Moore [148]) был предложен псевдополиномиальный алгоритм. Точный метод для данного случая с несколькими приборами $R \mid\mid \sum w_j U_j$, основанный на декомпозиции Данцига-Вулфа, представлен в работе Чена и Пауэла (Chen, Powell [106]).

Баптист и др. (Baptiste et al. [85]) предложили алгоритм для оптимального решения общего случая без весов с временами поступления $1 \mid r_j \mid \sum U_j$. Данному алгоритму удалось решить более 99% тестовых примеров со 100 требованиями, а также 80% тестовых примеров со 160 требованиями (при ограничении времени в 1 час на решение каждого примера).

В литературе появились точные методы решения общего случая задачи с весами и временами поступления $1 \mid r_j \mid \sum w_j U_j$. Сначала Баптист и др. (Baptiste et al. [83]) представили алгоритм, основанный на методе программирования в ограничениях. Данный алгоритм также может применяться для решения многоприборной задачи, когда приборы идентичны ($P \mid r_j \mid \sum w_j U_j$). Затем алгоритм, основой которого является определение *главной последовательности* (*master sequence*), был предложен Дозер-Пере и Сево (Dauzère-Pérès, Sevaux [112]). В данной работе задача формулируется как оригинальная задача ЧЦЛП и решается методом множителей Лагранжа. Другой точный метод, основанный на методе множителей Лагранжа, был разработан Периди и др. (Péridy et al. [180]). С помощью данного алгоритма было оптимально решено 84,4% тестовых примеров со 100 требованиями (на решение каждого примера отводился 1 час). Для решения примеров большей размерности Сево и Дозер-Пере (Sevaux, Dauzère-Pérès [200]) предложили генетический алгоритм локального поиска.

4.3 Формулировка задачи $1 \mid r_j \mid \sum w_j U_j$ как задачи ЧЦЛП

Задача $1 \mid r_j \mid \sum w_j U_j$ может быть сформулирована как задача ЧЦЛП. Пусть булева переменная x_j принимает значение 1, если требование $j \in N$ обслуживается вовремя и 0 – если требование j запаздывает. Важно заметить, что запаздывающие требования могут быть обслужены как угодно поздно без влияния на целевую функцию задачи, поэтому нам не обязательно знать время начала/окончания обслуживания запаздывающих требований в расписании. Пусть переменная s_j обозначает время начала обслуживания требования $j \in N$, а C_j – время окончания обслуживания требования j в расписании. Переменная y_{ij} принимает значение 1, если требование $i \in N$ предшествует

обслуживанию требования $j \in N$, $j \neq i$, в расписании и 0 – в противном случае. Теперь мы можем записать формулировку задачи, как задачи ЧЦЛП:

$$(\text{BP}) : \sum_{j \in N} w_j(1 - x_j) \longrightarrow \min \tag{4.27}$$

$$s_j + p_j x_j - C_j = 0, \quad \forall j \in N; \tag{4.28}$$

$$e_i - s_j + M y_{ij} \leq M, \quad \forall i, j \in N, i \neq j; \tag{4.29}$$

$$r_j \leq s_j, \quad \forall j \in N; \tag{4.30}$$

$$d_j \geq C_j, \quad \forall j \in N; \tag{4.31}$$

$$y_{ij} + y_{ji} - x_j \leq 0, \quad \forall i, j \in N, i \neq j; \tag{4.32}$$

$$x_i + x_j - y_{ij} - y_{ji} \leq 1, \quad \forall i, j \in N, i < j; \tag{4.33}$$

$$x_j \in \{0, 1\}, \quad \forall j \in N; \tag{4.34}$$

$$y_{ij} \in \{0, 1\}, \quad \forall i, j \in N, i \neq j; \tag{4.35}$$

$$s_j \geq 0, C_j \geq 0, \quad \forall j \in N. \tag{4.36}$$

Здесь ограничения (4.28) связывают переменные начала и окончания обслуживания требований. Ограничения предшествования (4.29) гарантируют, что обслуживание требования $j \in N$ начинается после окончания обслуживания требования $i \in N$, если $y_{ij} = 1$. Здесь M – некоторое большое число, обеспечивающее выполнение ограничений при $y_{ij} = 0$. Например, M можно положить равным разнице между максимальным директивным сроком и минимальным временем поступления. Ограничения (4.30) и (4.31) обеспечивают соблюдение времён поступлений и директивных сроков. Заметим, что в задаче $1 \mid r_j \mid \sum w_j U_j$ директивные сроки имеют характер deadline. Ограничения (4.32) и (4.33) связывают переменные x и y: первые из них обеспечивают равенство переменных y_{ij} и y_{ji}, $i \in N$, нулю, если требование $j \in N$ запаздывает ($x_j = 0$); вторые гарантируют, что если оба требования из пары (i, j) обслуживаются вовремя, то одно из этих требований предшествует другому.

Формулировка (BP) не может быть эффективно решена на практике стандартными методами целочисленного программирования. Во-первых, в ней содержится достаточно большое число переменных и ограничений ($O(n^2)$). Во-вторых, в ней присутствуют так называемые *"Big-M"* ограничения (4.29), линейная релаксация которых очень сильно удалена от выпуклой оболочки множества целочисленных решений (BP).

Переформулируем задачу (BP) в виде "гибридной" формулировки (GP):

$$(\text{HP}) : \sum_{j \in N} w_j(1 - x_j) \longrightarrow \min \tag{4.37}$$

$$\mathcal{R}_{D(x)}; \tag{4.38}$$
$$\texttt{disjunctive}(x); \tag{4.39}$$
$$x \in \{0,1\}^n. \tag{4.40}$$

Здесь ограничение (4.39) выполняется, когда множество требований $J = \{j \in N \mid x_j = 1\}$ может быть обслужено на одном приборе без нарушений времён поступления и директивных сроков. Ограничения (4.38) являются релаксацией ограничений (4.39).

Формулировка (HP) может быть решена в рамках "гибридной" схемы, представленной в разделе 4.1. Ограничениям (4.8) общей модели (GP) здесь соответствуют ограничение (4.39), которое моделируется линейными ограничениями достаточно громоздко и неэффективно, что существенно осложняет решение задачи ЧЦЛП. Например, в модели (BP) данное ограничение было описано с помощью ограничений (4.28)–(4.36) с привлечением дополнительных переменных y, C и s. Изначально в задаче нет других ограничений, кроме (4.39), однако, как это уже было сказано выше, отсутствие "сложных" ограничений не должно сильно влиять на оптимальное значение линейной релаксации задачи. Поэтому в качестве "легких" ограничений (4.6) предлагается релаксация (4.38). Конкретные реализации релаксации (4.38) будут представлены в разделе 4.3.1.

Одним из важнейших вопросов при создании "гибридного" алгоритма является порядок проверки "сложных" ограничений и отсечение недопустимых решений. Ограничение (4.39) является стандартным глобальным ограничением метода ПвО. Поэтому проверка допустимости текущего целочисленного решения по отношению к ограничению (4.39) может быть осуществлена решением ЗВСО по методу ПвО. Данное ограничение также может быть проверено комбинаторными алгоритмами решения задачи минимизации максимального временно́го смещения, так данная проверка представляет собой задачу распознавания для задачи $1 \mid r_j \mid L_{\max}$, о которой говорилось в предыдущих главах. Подробнее на алгоритмах проверки ограничения (4.39) и построении отсечений мы остановимся в разделе 4.4.

4.3.1 Дополнительные ограничения

Как было сказано в разделе 4.1 задача вида (GP) может быть успешно решена в рамках рассмотренной "гибридной" схемы только, если число решений допустимых для релаксации (MIP) и недопустимых для полной задачи (GP) невелико. В нашем же случае "легкие" ограничения (4.6) вообще отсутствуют

и очевидно, что существует много решений допустимых для (MIP) и недопустимых для (GP).

Выходом в данной ситуации является добавление релаксации ограничения `disjunctive` в задачу (MIP) к ограничениям (4.13). Для рассматриваемого ограничения (4.39) тривиальная релаксация была предложена в [90, 134]:

$$\sum_{j \in N} p_j x_j \leq \max_{j \in N} d_j - \min_{j \in N} r_j. \tag{4.41}$$

Ограничение (4.41) основано на таком наблюдении, что сумма продолжительностей обслуживания всех требований, которые обслуживаются вовремя, не должна превышать разницы между максимальным директивным сроком и минимальным временем поступления.

Однако, для решения примеров большой размерности релаксация (4.41) не является приемлемой, так как достаточно большое число недопустимых решений x задачи удовлетворяет данному ограничению. В этом разделе мы предложим другую релаксацию ограничения (4.39).

Отправным моментом нам послужит тот факт, что сумма времён обслуживания всех требований, которые должны быть выполнены в интервале между временем поступления одного требования и директивным сроком другого $[r_i, d_j]$ не может превышать длины данного интервала.

Лемма 4.1 *Пусть для двух требований $i, j \in N$ выполняется $r_i < d_j$, и пусть $S_{ij} = \{l \in N \mid r_l \geq r_i, d_l \leq d_j\}$. Тогда вектор x, удовлетворяющий ограничению `disjunctive`, удовлетворяет следующему ограничению:*

$$\sum_{l \in S_{ij}} p_l x_l \leq d_j - r_i. \tag{4.42}$$

Доказательство. Докажем от противного. Пусть решение x не удовлетворяет ограничению (4.42). Тогда сумма продолжительностей обслуживания требований из S_{ij}, обслуживаемых вовремя, превышает длину интервала $[r_i, d_j]$. Но все такие требования должны обслуживаться внутри данного интервала. То есть нарушается ограничение, согласно которому в каждый момент времени может обслуживаться не более одного требования, и x не удовлетворяет ограничению `disjunctive`. □

На рисунке 4.1 представлен пример интервала $[r_i, d_j]$ и требования, которые могут быть выполнены вовремя только в данном интервале. Сумма длин темных прямоугольников (продолжительностей обслуживания) для требований, выполненных вовремя, не должна превышать длины интервала.

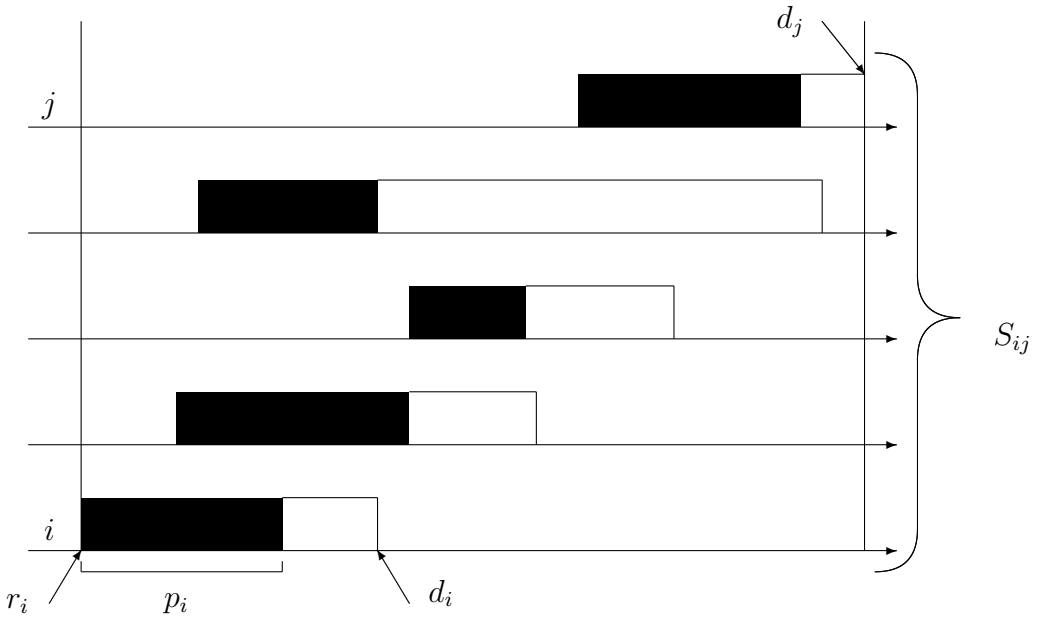

Рис. 4.1: Отрезок $[r_i, d_j]$ и требования, обслуживаемые в нем

Для усиления ограничения (4.41) мы рассмотрим не только те требования, которые должны быть полностью обслужены в интервале $[r_i, d_j]$, но и требования, хотя бы частично выполняемые в данном интервале. Через χ_+ будем обозначать положительную часть величины χ: $\chi_+ = \max\{\chi, 0\}$. Пусть $\alpha_{li} = (r_i - r_l)_+$ и $\beta_{jl} = (d_l - d_j)_+$.

Лемма 4.2 *Пусть для двух требований $i, j \in N$ выполняется $r_i < d_j$. Тогда вектор x, удовлетворяющий ограничению* `disjunctive`*, удовлетворяет ограничению*

$$\sum_{l \in N} \min[d_j - r_i, (p_l - \max\{\alpha_{li}, \beta_{jl}\})_+] x_l \leq d_j - r_i. \tag{4.43}$$

Доказательство. Для каждого требования l мы покажем, что по крайней мере $\min[d_j - r_i, (p_l - \max\{\alpha_l^{ij}, \beta_l^{ij}\})_+]$ из его общей продолжительности обслуживания p_l должно находиться в интервале $[r_i, d_j]$, если требование l обслуживается вовремя.

Если $r_l \geq r_i$ или $\alpha_{li} = 0$, то максимальное количество продолжительности обслуживания, приходящееся на время после d_j составляет $(d_l - d_j)_+ = \max\{\alpha_{li}, \beta_{jl}\}$. Поэтому минимальное время обслуживания требования l внутри интервала $[r_i, d_j]$ равняется $(p_l - \max\{\alpha_{li}, \beta_{jl}\})_+$, что меньше или равно разнице $d_j - r_i$. Случай, когда $d_l \leq d_j$ или $\beta_{jl} = 0$ симметричен и показывается аналогично.

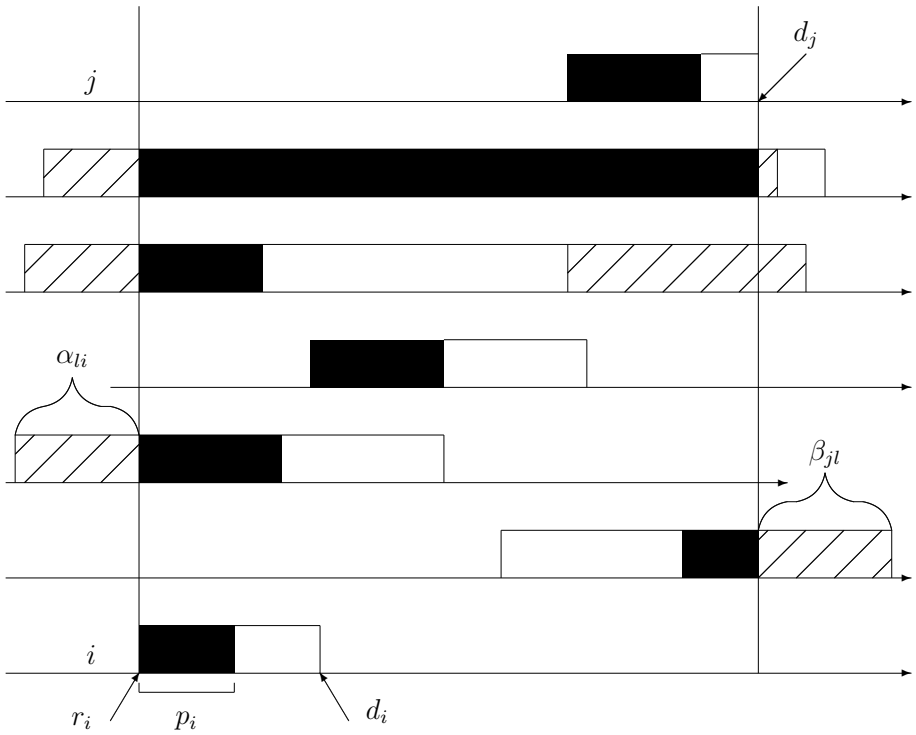

Рис. 4.2: Отрезок $[r_i, d_j]$ и требования, частично или полностью обслуживаемые в этом интервале.

Иначе интервал $[r_i, d_j]$ находится строго в интервале $[r_l, d_l]$, или, что эквивалентно, $\alpha_{li} > 0$ и $\beta_{jl} > 0$. Пусть $\beta_{jl} \geq \alpha_{li}$ (обратный случай симметричен). Тогда, если $p_l - \beta_{jl} < d_j - r_i$, то максимальное количество продолжительности обслуживания требования l, лежащее за пределами интервала $[r_i, d_j]$, составляет β_{jl}. Поэтому минимальное время обслуживания требования l внутри интервала равняется $p_l - \beta_{jl} = p_l - \max\{\alpha_{li}, \beta_{jl}\}$. Соответственно, если $p_l - \beta_{jl} \geq d_l - r_i$, то требование l обслуживается в течение всего интервала, т.е. занимает всё время $d_j - r_i$. □

На рисунке 4.2 представлен пример интервала $[r_i, d_j]$ и требования, которые, будучи выполненными вовремя, по крайней мере частично должны обслуживаться в данном интервале. Темными прямоугольниками на рисунке показаны части продолжительностей обслуживания, которые должны обслуживаться в интервале $[r_i, d_j]$. Штрихованные части продолжительностей обслуживания могут находится за пределами интервала. Сумма длин темных прямоугольников (продолжительностей обслуживания) для требований, выполненных вовремя, не должна превышать длины интервала.

Ограничения (4.43) мы будем использовать в качестве ограничений (4.38) задачи (HP).

4.4 Генерация отсечений

В данном разделе мы подробно остановимся на вопросах проверки допустимости решения и построения отсечений в алгоритме ветвей и отсечений, предназначенном для решения рассматриваемой задачи.

Пусть дано целочисленное (битовое) решение $\bar{x} \in \{0,1\}^n$ задачи ЧЦЛП, состоящей из целевой функции (4.37), также ограничений (4.38) и (4.40). Обозначим множество требований $\bar{J} = \{j \in N \mid \bar{x}_j = 1\}$. Необходимо проверить, существует ли расписание $\pi' \in \Pi(\bar{J})$, в котором требования обслуживаются без нарушения времён поступления и директивных сроков. То есть максимальное временно́е смещение расписания π' должно быть меньше или равно нулю. Поэтому для проверки допустимости решения $\bar{x}$ достаточно решить задачу $1 \mid r_j \mid L_{\max}$ для множества требований $\bar{J}$. Если при этом минимальное временно́е смещение полученного примера больше нуля, то мы можем построить стандартное отсечение вида (4.26):

$$\sum_{j \in \bar{J}} x_j \leq \mid \bar{J} \mid -1. \tag{4.44}$$

Используя термины, введенные в разделе 3.4, можно заметить, что решение $\bar{x}$ допустимо тогда и только тогда, когда множество требований $\bar{J}$ допустимо по отношению к константе 0. В случае недопустимости $\bar{J}$ мы можем найти недопустимое подмножество $S \subseteq \mid J \mid$ с помощью модифицированного алгоритма Карлиера (алгоритм 3.7). Тогда на основе подмножества S может быть построено более сильное отсечение

$$\sum_{j \in S} x_j \leq \mid S \mid -1. \tag{4.45}$$

Теперь мы представим другой способ генерации отсечений. Ограничения (4.43) могут быть обобщены следующим образом. Заметим, что данные неравенства задаются для отрезков, левым концом которых является время поступления какого-либо требования, а правым концом – директивный срок. Используя алгоритмы фильтрации ПвО, мы можем уменьшить "временно́е окно" некоторых требований (т.е. промежуток времени, в котором требование может быть обслужено), тем самым изменив определённые времена поступления и директивные сроки. Идея обобщения ограничений (4.43) заключается в использовании изменённых значений времён поступления и директивных сроков. Заметим однако, что "временно́е окно" любого требования может быть уменьшено только при условии, что другие требования, вызвавшие данное изменения, обслуживаются в расписании вовремя.

В данном разделе мы рассмотрим случай, когда время поступления некоторого требования k может быть увеличено, если все требования из определенного множества Ω обслуживаются вовремя. Случай, когда уменьшается некоторый директивный срок, полностью симметричен и здесь не представлен. Мы также не рассматриваем случай, когда одновременно изменяются несколько времён поступления или директивных сроков, или их комбинация.

Предоставим возможность заинтересованному читателю провести соответствующие рассуждения...

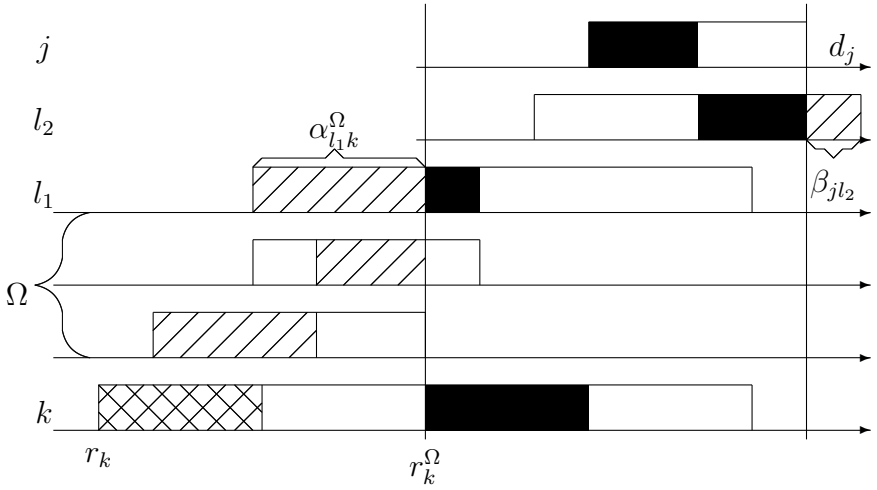

Рис. 4.3: Пример обобщенного ограничения (4.46)

Утверждение 4.3 представляет и обосновывает обобщенное ограничение для рассматриваемого случая. Пример данного ограничения графически изображен на рисунке 4.3. Закрашенные области требований здесь обозначают коэффициенты при переменных, соответствующих данным требованиям. Обозначим $a_l(E)$, $l \in N$, и $b(E)$ как, соответственно, коэффициент при переменной x_l в правой части и значение правой части неравенства E. Пусть также $A_x(E)$ является значением левой части неравенства E для вектора переменных x: $A_x(E) = \sum\limits_{l \in N} a_l(E) x_l$.

Лемма 4.3 *Пусть заданы такие множество требований $\Omega \subset N$ и требование $k \in N \setminus \Omega$, что k может быть обслужено только с момента времени r^{Ω}_k, $r^{\Omega}_k > r_k$, если все требования из Ω выполняются вовремя. Положим также $\alpha^{\Omega}_{lk} = (r^{\Omega}_k - r_l)_+$. Тогда вектор x, удовлетворяющий ограничению* `disjunctive`*, удовлетворяет неравенству*

$$\sum_{l \in N \setminus \Omega \setminus \{k\}} \min\left[d_j - r^{\Omega}_k, (p_l - \max\{\alpha^{\Omega}_{lk}, \beta_{jl}\})_+\right] x_l + (p_k - \beta_{jk})_+ x_k \le d_j - r^{\Omega}_k + (r^{\Omega}_k - r_k)(\mid \Omega \mid - \sum_{o \in \Omega} x_o), \tag{4.46}$$

для каждого требования $j \in N \setminus \Omega$, $d_j > r_k^\Omega$.

Доказательство. Рассмотрим неравенство E вида (4.46) для множества Ω, требования k и некоторого требования $j \in N \setminus \Omega$, $d_j > r_k^\Omega$.

Пусть $\sum\limits_{o \in \Omega} x_o \leq |\ \Omega\ | - 1$. Обозначим через E' неравенство (4.43) для интервала $[r_k, d_j]$. Согласно Лемме 4.2, любое допустимое решение x удовлетворяет неравенству E', т.е. $\sum\limits_{l \in N} a_l(E')x_l \leq b(E')$. Для доказательства данного случая мы покажем, что $b(E') \leq b(E)$ и $a_l(E) \leq a_l(E')$, $\forall l \in N$.

Имеем $b(E) \geq d_j - r_k = b(E')$. Из $r_k^\Omega \geq r_k$ следует $\alpha_{lk}^\Omega \geq \alpha_{lk}$ и $a_l(E) \leq a_l(E')$, $\forall l \in N \setminus \Omega \setminus \{k\}$. Затем, $a_l(E) = 0 \leq a_l(E')$, $\forall l \in \Omega$. Наконец, $a_k(E) = a_k(E')$, и $a_l(E) \leq a_l(E')$, $\forall l \in N$.

Пусть теперь $\sum\limits_{o \in \Omega} x_o = |\ \Omega\ |$. Тогда время поступления требования k может быть увеличено до r_k^Ω. Обозначим через E'' неравенство (4.43) для интервала $[r_k^\Omega, d_j]$. Идея доказательства такая же, как и в предыдущем случае.

Имеем $b(E) = b(E'')$ и $a_l(E) = a_l(E'')$, $\forall l \in N \setminus \Omega$. Затем, $a_l(E) = 0 \leq a_l(E'')$, $\forall l \in \Omega$. Снова получаем $a_l(E) \leq a_l(E')$, $\forall l \in N$. □

Как можно заметить, число неравенств (4.46) экспоненциально (от количества требований), и мы не можем использовать все такие ограничения в качестве релаксации (4.38) ограничения `disjunctive`. Но данные обобщённые неравенства можно использовать, как отсечения недопустимых решений. Для этого требуется разработать алгоритм отсечения, который для заданного решения $\bar{x}$ находит неравенство вида (4.46), которому $\bar{x}$ не удовлетворяет.

Любое решение $\bar{x}$ удовлетворяет ограничениям (4.43). Используя этот факт, сначала мы выделим случаи, когда $\bar{x}$ заведомо не может нарушать ни одно из неравенств (4.46). Следующие три утверждения раскрывают такие случаи.

Лемма 4.4 *Если целочисленный вектор-решение* $\bar{x}$ *нарушает неравенство* E *вида (4.46) для множества требований* $\Omega \subset N$ *и требований* $k, j \in N \setminus \Omega$, *и существует такое требование* $l \in \Omega$, *что* $\bar{x}_l = 0$, *тогда* $\bar{x}$ *также нарушает по крайней мере одно из неравенств (4.43).*

Доказательство. Обозначим через E' неравенство (4.43) для интервала $[r_k, d_j]$. Так как ограничение E нарушается вектором $\bar{x}$, то выполняется $A_{\bar{x}}(E) > b(E)$. Для доказательства утверждения мы покажем, что значение правой части неравенства E больше или равно значению правой части

неравенства E' и значение левой части E меньше или равно левой части ограничения E'.

Имеем $\sum_{o\in\Omega} \bar{x}_o \leq \mid \Omega \mid -1$, поэтому $b(E) \geq d_j - r_k = b(E')$. Из $r_k^\Omega > r_k$ следует $\alpha_{lk}^\Omega \geq \alpha_{lk}$ и $a_l(E) \leq a_l(E')$, $\forall l \in N \setminus \Omega \setminus \{k\}$. Затем, $a_l(E) = 0 \leq a_l(E')$, $\forall l \in \Omega$. Наконец, $a_k(E) = a_k(E')$ и $A_{\bar{x}}(E) \leq A_{\bar{x}}(E')$. □

Лемма 4.5 *Если целочисленный вектор-решение $\bar{x}$ нарушает неравенство E вида (4.46) для множества требований $\Omega \subset N$, требования $k \in N \setminus \Omega$ и такого требования $j \in N \setminus \Omega$, что $d_j \leq d_k - p_k$, тогда $\bar{x}$ также нарушает по крайней мере одно из неравенств (4.43).*

Доказательство. Пусть $\sum_{o\in\Omega} \bar{x}_o = \mid \Omega \mid$, в противном случае доказательство следует из леммы 4.4. Мы рассмотрим следующие три случая.

1. Существует требование $i \in N \setminus \Omega$, $r_i < r_k^\Omega$, $r_i + p_i > r_k^\Omega$, $d_i \leq d_j$, $\bar{x}_i = 1$. Обозначим через E' неравенство (4.43) для интервала $[r_i, d_j]$. Имеем $b(E') = b(E) + (r_k^\Omega - r_i)$. Для доказательства случая 1 мы покажем, что $A_{\bar{x}}(E')$ больше чем $A_{\bar{x}}(E)$ как минимум на $r_k^\Omega - r_i$.

Из $r_i < r_k^\Omega$ следует $\alpha_{ik} \leq \alpha_{lk}^\Omega$ и $a_l(E) \leq a_l(E')$, $\forall l \in N \setminus \Omega \setminus \{k\}$. Затем, $a_k(E) = 0 = a_k(E')$, так как $d_j \leq d_k - p_k$, и $a_l(E) = 0 \leq a_l(E')$, $\forall l \in \Omega$. Более того, из $d_i \leq d_j$ следует $a_i(E) = p_i - (r_k^\Omega - r_i)$, и $a_i(E') = p_i$. Следовательно, $a_i(E) - a_i(E') = r_k^\Omega - r_i$ и $A_{\bar{x}}(E') - A_{\bar{x}}(E) \geq r_k^\Omega - r_i$.

2. Случай 1 не выполняется и существует не более одного такого требования $i \in N \setminus \Omega$, что $r_i < r_k^\Omega$, $r_i + p_i > r_k^\Omega$, $d_i - p_i < d_j$, $\bar{x}_i = 1$. Пусть $m \in N \setminus \Omega$ – такое требование с наименьшим временем поступления, что $r_m \geq r_k^\Omega$, $r_m < d_j$, $a_m(E) > 0$, $\bar{x}_m = 1$. Такое требование m должно существовать, иначе решение $\bar{x}$ не нарушало бы ограничение E, так как переменная, соответствующая не более одному требованию i, $\bar{x}_i = 1$, имела бы ненулевой коэффициент в E (любой коэффициент не может быть больше $b(E)$). Обозначим через E'' неравенство (4.43) для $[r_m, d_j]$. Имеем $b(E'') = b(E) - (r_m - r_k^\Omega)$. Для доказательства случая 2 мы покажем, что $A_{\bar{x}}(E'')$ меньше, чем $A_{\bar{x}}(E)$ как максимум на $r_m - r_k^\Omega$.

Мы имеем $r_l + p_l < r_k^\Omega$ или $r_l \geq r_m$ для всех требований из N за исключением не более одного требования i, поэтому $a_l(E'') = a_l(E)$, $\forall j \in N \setminus \{i\}$. Из $\alpha_{im} - \alpha_{ik}^\Omega = r_m - r_k^\Omega$ следует $a_i(E) - a_i(E'') \leq r_m - r_k^\Omega$ и $A_{\bar{x}}(E) - A_{\bar{x}}(E'') \leq r_m - r_k^\Omega$.

3. Случай 1 не выполняется и существуют как минимум два требования $i_1, i_2 \in N \setminus \Omega$, $r_{i_t} < r_k^\Omega$, $r_{i_t} + p_{i_t} > r_k^\Omega$, $d_{i_t} - p_{i_t} < d_j$, $\bar{x}_{i_t} = 1$, $t = 1, 2$. Так как случай 1 не выполняется, мы имеем $d_{i_1} > d_j$ и $d_{i_2} > d_j$. Без потери общности

будем полагать, что $d' = d_{i_1} \le d_{i_2}$ и $r' = \max\{r_{i_1}, r_{i_2}\}$. Обозначим через E^* неравенство (4.43) для интервала $[r', d']$. Имеем $b(E^*) = b(E) + (r_k^\Omega - r') + (d' - d_j)$. Для доказательства случая 3 мы покажем, что $A_{\bar{x}}(E^*)$ больше чем $A_{\bar{x}}(E)$ как минимум на $(r_k^\Omega - r') + (d' - d_j)$.

Из $r' < r_k^\Omega$ и $d_j < d'$ следует $a_l(E^*) \ge a_l(E)$, $\forall l \in N$. Пусть сначала $r' = r_{i_2}$, тогда $\beta_{i_1 i_1} = 0$, $\alpha^\Omega_{i_1 k} - \alpha_{i_1 i_2} = r_k^\Omega - r'$, и $a_{i_1}(E^*) - a_{i_1}(E) \ge r_k^\Omega - r'$. Также, $\beta_{i_2 i_2} = 0$, $\beta_{j i_2} - \beta_{i_1 i_2} = d' - d_j$, и $a_{i_2}(E^*) - a_{i_2}(E) \ge d' - d_j$.

Пусть теперь $r' = r_{i_1}$, тогда $a_{i_1}(E^*) = p_{i_1}$ и $a_{i_1}(E^*) - a_{i_1}(E) = \max\{r_k^\Omega - r', d' - d_j\}$. Также, $\alpha^\Omega_{i_2 k} - \alpha_{i_2 i_1} = r_k^\Omega - r'$, $\beta_{j i_2} - \beta_{i_1 i_2} = d' - d_j$, поэтому $a_{i_2}(E^*) - a_{i_2}(E) \ge \min\{r_k^\Omega - r', d' - d_j\}$. Следовательно, $A_{\bar{x}}(E^*) - A_{\bar{x}}(E) \ge (r_k^\Omega - r') + (d' - d_j)$. □

Будем обозначать $C_S = r_S + p_S$ для произвольного множества требований S.

Лемма 4.6 *Если целочисленный вектор-решение $\bar{x}$ нарушает неравенство E вида (4.46) для множества требований $\Omega \subset N$, требований $k, j \in N \backslash \Omega$, и существует такое подмножество $J' \subseteq \bar{J}$, что $d_{J'} \le \max\{d_j, d_k\}$, $C_{J'} \ge r_k^\Omega$, $r_{J'} \le r_k$, и $C_{J''} \le C_{J'}$, $\forall J'' \subset J'$, тогда $\bar{x}$ также нарушает по крайней мере одно из неравенств (4.43).*

Доказательство. Пусть $\sum_{o \in \Omega} \bar{x}_o = \mid \Omega \mid$, в противном случае доказательство следует из леммы 4.4. Также $d_j > d_k - p_k$, иначе доказательство следует из леммы 4.5. Обозначим $d' = \max\{d_k, d_j\}$.

Обозначим через E^* неравенство (4.43) для интервала $[r_{J'}, d']$. Имеем $b(E') = b(E) + (r_k^\Omega - r_{J'}) + (d' - d_j)$. Для доказательства утверждения мы покажем, что $A_{\bar{x}}(E')$ больше чем $A_{\bar{x}}(E)$ как минимум на $r_k^\Omega - r_{J'} + (d' - d_j)$.

Из $d' \ge d_{J'}$ следует $a_l(E) \le (p_l - \alpha^\Omega_{lk})_+$, $a_l(E') = p_l$, $\forall l \in J'$. Также, $a_k(E') - a_k(E) = d' - d_j$. Тогда, из $a_l(E) \le a_l(E')$, $\forall l \in N$, мы имеем $A_{\bar{x}}(E') - A_{\bar{x}}(E) \ge p_{J'} - \sum_{l \in J'}(p_l - \alpha^\Omega_{lk})_+ + (d' - d_j)$. Следовательно, достаточно показать, что

$$\sum_{l \in J'} (p_l - \alpha^\Omega_{lk})_+ \le p_{J'} - (r_k^\Omega - r_{J'}). \tag{4.47}$$

Докажем данное неравенство от противного. Пусть $J'' = \{l \in J' : p_l - \alpha^\Omega_{lk} \ge 0\}$ и $r_{l'} = r_{J''}$. Предположим, что (4.47) не выполняется, тогда

$$C_{J'} = r_{J'} + p_{J'} < \sum_{l \in J''} (p_l - \alpha^\Omega_{lk}) + r_k^\Omega = p_{l'} - (r_k^\Omega - r_{l'})_+ + r_k^\Omega +$$

$$\sum_{l \in J'' \backslash \{l'\}} (p_l - (r_k^\Omega - r_{l'})_+) \le r_{l'} + p_{l'} + \sum_{l \in J'' \backslash \{l'\}} p_l = r_{J''} + p_{J''} = C_{J''},$$

что противоречит условию утверждения. □

Теперь мы подробнее остановимся на алгоритме отсечения. Для целочисленного решения $\bar{x}$ данный алгоритм находит одно из неравенств (4.46), нарушенное $\bar{x}$ и основанное на увеличении времени поступления некоторого требования алгоритмом фильтрации "Edge-Finding" (алгоритм 3.3), если такое неравенство существует. Заметим, что алгоритм не гарантирует нахождение произвольного неравенства (4.46), которое нарушается решением $\bar{x}$, так как существуют и другие алгоритмы фильтрации для ограничения `disjunctive`.

Идея алгоритма отсечения заключается в следующем. В качестве основы используется алгоритм фильтрации "Edge-Finding" (алгоритм 3.3). Каждый раз, когда данный алгоритм изменяет время поступления некоторого требования k в соответствии с правилом (3.4)–(3.5), происходит проверка на существование такого неравенства E вида (4.46) для множества Ω, требования k, и некоторого требования $j \in N \setminus \Omega$, $d_j > d_k - p_k$, что $\bar{x}$ нарушает E. Такая проверка требует $O(n^2)$ операций. Алгоритм фильтрации выполняется для множества требований $\bar{J} = \{j \in N : \bar{x}_j = 1\}$, так как, согласно лемме 4.4, неравенство E всегда выполняется для $\bar{x}$, если существует требование $l \in \Omega$, $\bar{x} = 0$.

Алгоритмы фильтрации "Edge-Finding" имеют сложность $O(n \log n)$ или $O(n^2)$ операций [84]. Поэтому сложность простого комбинированного алгоритма составляет $O(n^3 \log n)$ или $O(n^4)$. Здесь мы представим алгоритм отсечения сложности $O(n^3)$, детально описанный как алгоритм 4.4. В алгоритме предполагается, что требования множества $\bar{J}$ отсортированы по неубыванию времён поступления.

На итерации l внешнего цикла множество требований $\bar{J}$ разбивается на два подмножества $\Omega_{\leq}$ и $\Omega_{>}$ (строки 2-3), куда требования попадают в зависимости от отношения их директивных сроков к d_l. Поиск множества Ω будет осуществляться среди подмножеств $\Omega_{\leq,i} = \{j \in \Omega_{\leq} : r_j \geq r_i\}$. Для каждого требования $k \in \Omega_{>}$ мы будем проверять, возможно ли увеличить время поступления r_k до C. Значение r_k не может быть увеличено до значения, большего чем C, в следствие определения C (строка 4) и правила (3.5). Ниже по тексту мы покажем, почему не рассматриваются значения меньшие чем C.

В начале итерации k внутреннего цикла, если $d_k > d_l$, на строке 15, мы располагаем следующими значениями:

$$P = p_{\Omega_{\leq,k}}, \qquad H = \max_{i<k}\{r_i + p_{\Omega_{\leq,i}}\} = r_h + p_{\Omega_{\leq,h}} = C_{\Omega_{\leq,h}}.$$

Алгоритм 4.4 Алгоритм отсечения

for $l := 1$ to $\mid \bar{J} \mid$ **do**
  $\Omega_{\leq} := \{i \in \bar{J} : d_i \leq d_l\}$;
  $\Omega_{>} := \{i \in \bar{J} : d_i > d_l\}$;
  $P := p_{\Omega_{\leq}}$; $C := \max\limits_{\Omega' \subseteq \Omega_{\leq}} \{r_{\Omega'} + p_{\Omega'}\}$;
  $\alpha'_i := (C - r_i)_+$, $\forall i \in \Omega_{>}$;
  $\bar{d} := \{\}$; $\bar{d} \leftarrow \{d_i; d'_i = d_i - \max\{p_i, \alpha'_i\}; d''_i := d_i - p_i\}$, $\forall i \in \Omega_{>}$;
  отсортируем по неубыванию $\bar{d}$; $H := -\infty$;
  for $k := 1$ to $\mid \bar{J} \mid$ **do**
    if $d_k \leq d_l$ **then**
      if $H < r_k + P$ **then**
        $h := k$; $H := r_k + P$;
      end if
      $P := P - p_k$
    else
      if $(r_k + p_k + P > d_l$ **or** $H + p_k > d_l)$ **and** $H < C$ **then**
        if $r_k + p_k + P > d_l$ **then**
          $\Omega := \{k+1, \ldots, n\} \cap \Omega_{\leq}$;
        else
          $\Omega := \{h, \ldots, n\} \cap \Omega_{\leq}$;
        end if
        изменим значение d'_k на d_k в $\bar{d}$; пересортируем $\bar{d}$;
        CHECK$(\Omega, k, C, \alpha', \bar{d})$
        изменим значение d'_k на $d_k - \max\{p_k, \alpha'_k\}$ в $\bar{d}$; пересортируем $\bar{d}$;
      end if
    end if
  end for
end for

Алгоритм 4.5 Процедура проверки для алгоритма отсечения

procedure CHECK$(\Omega, k, C, \alpha', \bar{d})$
$z := 0$; $s := \sum\limits_{i \in \Omega_{>} \setminus \{k\}} (p_i - \alpha'_i) + p_k$; i – максимальный индекс в $\bar{d}$;
while $i > 1$ **and** $\bar{d}_i > d_k - p_k$ **do**
  if $s > \bar{d}_i - C$ **then**
    $\bar{x}$ нарушает неравенство (4.46), для множества Ω, для требования k, и требования j, $d_j = \bar{d}_i$; **end algorithm**;
  end if
  $i := i - 1$; $s := s - z \cdot (\bar{d}_{i+1} - \bar{d}_i)$;
  if $\bar{d}_i$ соответствует некоторому d'_j **then**
    $z := z + 1$;
  else if $\bar{d}_i$ соответствует некоторому d''_j **then**
    $z := z - 1$;
  end if
end while
end procedure

Рассмотрим случай, когда $H = C$ и условие (3.4) выполняется для требования k и некоторого множества $\Omega_{\leq,i}$. Заметим, что $r_k^\Omega \geq r_{\Omega_{\leq,h}}$, и $r_k^\Omega \leq C$ выполняется по определению C. Тогда, $d_{\Omega_{\leq,h}} \leq d_k$, в противном случае из (3.4) мы бы получили $p_{\Omega_\leq} + p_k > d_{\Omega_\leq \cup \{k\}} - r_{\Omega_\leq \cup \{k\}}$, и $\bar{x}$ нарушало бы неравенство (4.43) для интервала $[r_{\Omega_\leq \cup \{k\}}, d_{\Omega_\leq}]$. Поэтому множество $\Omega_{\leq,h} \subset \bar{J}$ может играть роль множества J' в условии леммы 4.6, и тогда, согласно данному утверждению, $\bar{x}$ не может нарушать неравенство (4.46) для множества $\Omega_{\leq,i}$, требования k, и любого требования j. Следовательно, имеет смысл рассматривать только случай, когда $H < C$.

На строке 15 условие (3.4) проверяется для пары $\Omega_{\leq,k}$, k ($r_k + p_k + P > d_l$), а также для пары $\Omega_{\leq,h}$, k ($H + p_k > d_l$). Если одно из этих неравенств выполняется, то мы располагаем множеством Ω, получаемом на строке 17 или 19. Так как $H < C$, то значение $C = \max\limits_{\Omega' \subseteq \Omega_\leq} \{C_{\Omega'}\}$ достигается на некотором множестве $\Omega_{\leq,i}$, $i > k$. Поэтому, $\Omega_{\leq,i} \subseteq \Omega_{\leq,k} \subseteq \Omega_{\leq,h}$ и в обоих случаях $r_k^\Omega = C$.

Далее, располагая парой (Ω, k), мы проверяем, существует ли неравенство (4.46) для Ω, k, и некоторого требования $j \in N$, которое не выполняется для $\bar{x}$. Мы покажем, как это сделать за линейное число операций.

Перед внутренним циклом мы уже знаем, что в случае выполнения условий "Edge-Finding" время поступления r_k может быть увеличено до $r_k^\Omega = C$, $\forall k \in \Omega_>$. Поэтому, в любом искомом неравенстве E вида (4.46) $a_i(E) = 0$, $\forall i \in \Omega_\leq$, так как $C - r_i > p_i$. Затем, перед внутренним циклом нам известны значения α_{ik}^Ω для всех $i \in \Omega_>$, $i \neq k$. Данные значения сохраняются в векторе α'_i на строке 5.

Обозначим через E' неравенство (4.46) для множества Ω, некоторого фиксированного требования k, и требования $j \in \Omega_>$, $d_j = d_{\Omega_>}$. Тогда $a_i(E') = (p_i - \alpha'_i)_+$, $\forall i \in \Omega_> \setminus \{k\}$. На строке 2 процедуры проверки переменной s мы присваиваем значение $A_{\bar{x}}(E')$. Если правая граница d_j интервала $[C, d_{\Omega_>}]$ уменьшается, то коэффициент переменной x_i, $i \neq k$, не меняется, пока $\alpha'_i \geq \beta_{ji}$, т.е. пока $d_j \geq d'_i = d_i - \max\{p_i, \alpha'_i\}$. Затем коэффициент начинает уменьшаться пока не станет равным нулю, когда $d_j = d''_i = d_i - p_i$. Для переменной x_k мы имеем $d'_k = d_k$. На строке 6 для каждого требования $i \in \Omega_>$ все три значения $\{d_i, d'_i, d''_i\}$ добавляются в массив $\bar{d}$ и сортируются. На строке 21, когда известно требование k, мы изменяем значение d'_k и снова сортируем $\bar{d}$ в порядке неубывания директивных сроков. Заметим, что в данном случае сортировка требует только $O(n)$ операций, так как в $\bar{d}$ изменилось только одно значение. На строке массив $\bar{d}$ возвращается в изначальное состояние.

Используя массив $\bar{d}$, процедура CHECK (алгоритм 4.5) проверяет существование требуемого неравенства за линейное число операций. Процедура

последовательно уменьшает правую границу интервала и надлежащим образом изменяет сумму коэффициентов переменных s. Если для некоторой границы d_j сумма s оказалась больше чем длина интервала $[C, d_j]$, то требуемое неравенство найдено.

Так как сложность процедуры проверки составляет $O(n)$, общая сложность алгоритма отсечения равняется $O(n^3)$ операций.

4.5 "Гибридный" алгоритм ветвей и отсечений

Итак, для задачи $1 \mid r_j \mid \sum w_j U_j$ предлагается следующий алгоритм ветвей и отсечений, построенный по схеме, представленной в алгоритме 4.2. Задача ЦЛП, состоящая из целевой функции (4.37), а также ограничений (4.46) и (4.40), решается стандартным методом ветвей и границ. Каждое целочисленное решение данной задачи проверяется на допустимость. В случае недопустимости решения, происходит генерация отсечения, которое добавляется к ограничениям задачи ЦЛП. Если решение допустимо, а это значит, что найдено допустимое расписание, то данное расписание запоминается, как лучшее текущее решение.

Мы рассмотрим пять вариантов предлагаемого алгоритма. Каждый вариант использует отличающуюся от других стратегию проверки решения задачи ЦЛП на допустимость и генерации отсечений. Другие компоненты алгоритма одинаковы для каждого из вариантов.

Напомним, что мы можем использовать три подхода для генерации отсечений. Первый, обычный метод, заключается в использовании стандартных отсечений (4.44). Для их генерации нам требуется только информация о том, допустимо ли решение $\bar{x}$ или нет. Для получения данного ответа решается пример задачи $1 \mid r_j \mid L_{\max}$ со множеством требований $\bar{J}$ алгоритмом 3.2. Обозначим данный стандартный подход, как "ng" (в англоязычной литературе генерируемые здесь отсечения называются *"no-good"*).

Второй подход использует отсечения вида (4.45), построенные для недопустимого подмножества требований $S \subseteq \bar{J}$. Для генерации данных отсечений используется алгоритм 3.7. К сожалению, на практике данный алгоритм не всегда работает достаточно быстро, поэтому мы прерываем его исполнение, если количество узлов в дереве поиска превысило 1000. Второй подход обозначается "tng" (*tightened "no-good" cuts* – "уплотненные" отсечения "no-good").

Следующий подход детально представлен в разделе 4.4. Здесь в каче-

стве отсечений используются ограничения (4.46). Для их построения применяется алгоритм 4.4. Обозначим этот подход как “gti” (*generalized tightening inequalities* – обобщенные “уплотняющие” неравенства).

Заметим, что алгоритм 4.4 отсечения не может ничего сообщить о том, допустимо ли решение или нет, если искомое неравенство (4.46) не найдено. Также, модифицированный алгоритм Карлиера не в состоянии установить допустимо ли решение, если исполнение алгоритма прервано при достижении лимита узлов. Поэтому во всех вариантах алгоритма первый способ проверки допустимости всегда является “страхующим”. То есть, если другие способы не смогли дать ответ о допустимости решения, то используется первый подход.

Будут рассмотрены следующие 5 вариантов алгоритма: “ng”, “gti+ng”, “gti+tng+ng”, “tng+ng”, “tng+gti+ng”. Например, обозначение последнего варианта “tng+gti+ng” означает, что сначала выполняется модифицированный алгоритм Карлиера. Если алгоритм прерывается до завершения работы, то применяется алгоритм отсечения. Наконец, если последний не нашел требуемого неравенства, используется алгоритм Карлиера.

4.6 Экспериментальная оценка эффективности

В данном разделе мы экспериментально оценим эффективность предложенного алгоритма ветвей и отсечений для решения задачи $1 \mid r_j \mid \sum w_j U_j$. Также будет проведено его сравнение с другим алгоритмом, представленным в литературе.

Для проведения экспериментов были использованы три набора тестовых примеров. Первый набор примеров использовался в работе [83]. Данные примеры были получены следующим образом:

- продолжительности обслуживания были сгенерированы равномерно на отрезке $[p_{\min}, p_{\max}]$;
- веса были сгенерированы равномерно на отрезке $[1, w_{\max}]$;
- времена поступления были сгенерированы согласно нормальному распределению с параметрами $(0, \sigma)$, где σ зависит от *загрузки* машины, более подробно см. [83]; *загрузка* прибора здесь равняется отношению суммы всех продолжительностей обслуживания к разнице между максимальным директивным сроком и минимальным временем поступления $d_N - r_N$;

- запасы обслуживания ($m_j = d_j - r_j - p_j$) были сгенерированы равномерно на отрезке $[0, m_{\max}]$.

Были использованы следующие значения параметров генерации:

$(p_{\min}, p_{\max})$	(0,100), (25,75);
$m_{\max}$	50, 200, 350, 500, 650;
загрузка	1.0, 1.6, 2.2;
$w_{\max}$	1, 10, 100.

Для каждой четверки параметров и каждой размерности $n \in \{10, 20, \ldots, 100\}$ был сгенерирован один пример. Всего в тестовом наборе присутствуют 90 примеров для каждой размерности. Обозначим тестовые примеры из данного набора как "b-n".

Второй набор тестовых примеров взят из работы [112]. Для данных примеров продолжительности обслуживания были сгенерированы равномерно на отрезке $[0, 100]$. При заданном числе требований n, другие данные были получены следующим образом:

- веса были сгенерированы равномерно на отрезке $[1, w_{\max}]$;
- времена поступления были сгенерированы равномерно на отрезке $[0, K_1 n]$;
- директивные сроки были сгенерированы равномерно на отрезке $[r_j + p_j, r_j + p_j + K_2 n]$.

Были использованы следующие значения параметров генерации:

$w_{\max}$	10, 99
K_1	1, 5, 10, 20
K_2	1, 5, 10, 20.

Для каждой тройки параметров и каждой размерности $n \in \{20, 40, \ldots, 140\}$ набор содержит 10 примеров. Всего было сгенерировано 320 примеров для каждой размерности. Обозначим тестовые примеры из данного набора как "sn".

Мы признательны Филиппу Баптисту и Марку Сево за пересылку нам данных тестовых примеров.

Третий набор тестовых примеров был нами сгенерирован самостоятельно, используя похожую процедуру, как и при получении набора примеров "s". Разница заключается в том, что веса здесь зависят от продолжительностей

обслуживания. Каждый вес w_j, $j \in N$, генерировался равномерно в интервале $[\max\{1, p_j - \delta\}, \min\{99, p_j + \delta\}]$, где δ обозначает степень зависимости. Были использованы два значения параметра δ: 10 и 25. Для параметров K_1 и K_2 были использованы такие же значения: 1, 5, 10, 20. Для каждой тройки параметров и каждой размерности $n \in \{80, 100, 120\}$ набор содержит 10 примеров. Всего было сгенерировано 160 примеров для каждой размерности и каждого значения параметра δ. Обозначим тестовые примеры из данного набора как "d-δ-n".

Во время проведения экспериментов были получены значения следующих параметров:

P_{1h}, P_{1000s} – процент примеров, оптимально решенных, соответственно, за один час и за 1000 секунд (значения всех других параметров подсчитывались только для оптимально решенных в пределах временно́го лимита примеров);

T_{av}, T_{max} – среднее и максимальное время в секундах, необходимое для оптимального решения примера;

N_{av}, C_{av} – среднее число узлов в дереве поиска и среднее число отсечений, необходимое для оптимального решения примера;

$PMCE_{av}$ – средняя эффективность модифицированного алгоритма Карлиера, т.е. средняя разность в процентах между мощностью исходного множества требований $\bar{J}$ и мощностью полученного недопустимого подмножества $S \subseteq \bar{J}$: $PMCE = \frac{|\bar{J}|-|S|}{|\bar{J}|}$ (для случаев, когда алгоритм нашёл недопустимое подмножество, и лимит количества узлов не был превышен);

$PMCL$ – процент случаев, когда был превышен лимит узлов в дереве поиска в модифицированном алгоритме Карлиера.

Эксперименты были осуществлены на компьютере с процессором Pentium IV 2 ГГц. и памятью 512 Мб. Алгоритм был реализован с помощью языка моделирования *Mosel* [108]. Для решения задачи ЦЛП использовался пакет *XPress-MP* [126].

В таблице 4.1 представлены результаты исследования всех пяти вариантов алгоритма; "–" означает, что соответствующий эксперимент не был проведён. Мы не тестировали вариант "tng+gti+ng" на примерах из набора "b", так как при решении данных примеров вариантом "tng+ng" в модифицированном алгоритме Карлиера очень редко превышал предел узлов в дереве

Таблица 4.1: Результаты экспериментального исследования пяти вариантов алгоритма ветвей и отсечений

	“ng”		“gti+ng”		“gti+tng+ng”		“tng+ng”		“tng+gti+ng”	
Тест	P_{1h}	T_{av}	P_{1h}	T_{av}	P_{1h}	T_{av}	P_{1h}	T_{av}	P_{1h}	T_{av}
s-40	99.7%	2.5	100%	0.9	100%	0.6	100%	0.6	100%	0.6
s-60	98.8%	16.6	99.7%	6.0	100%	3.3	100%	3.5	100%	3.4
s-80	95.3%	50.3	98.8%	15.2	100%	15.3	100%	15.4	100%	11.8
s-100	–	–	99.4%	88.5	99.7%	46.0	99.7%	28.6	100%	33.0
s-120	–	–	–	–	97.8%	132.4	99.7%	115.7	99.7%	104.9
s-140	–	–	–	–	92.5%	127.2	95.9%	189.8	96.2%	196.2
b-40	100%	0.8	100%	0.8	100%	0.6	100%	0.5	–	–
b-50	98.9%	5.5	100%	3.0	100%	1.7	100%	1.5	–	–
b-60	96.7%	38.5	98.9%	5.9	100%	3.3	100%	3.5	–	–
b-70	94.4%	79.4	98.9%	27.7	100%	7.0	100%	6.7	–	–
b-80	97.8%	91.1	98.9%	25.9	100%	14.5	100%	49.4	–	–
b-90	87.8%	137.2	96.7%	41.5	98.9%	36.8	98.9%	26.3	–	–
b-100	–	–	–	–	100%	104.2	100%	91.6	–	–

поиска. Данный факт означает, что вариант “tng+gti+ng” имел бы очень схожие результаты для данных примеров. Сравнение показывает, что вариант “tng+gti+ng” является наилучшим для решения примеров из набора “s”, и вариант “tng+[gti+]ng” лучше всех справляется с примерами из набора “b”. Единственное исключение приходится на один из примеров набора “b80”. Данный пример был решен намного быстрее вариантом “gti+tng+ng”, что сказалась на общей статистике для всего набора “b80”.

В следующем эксперименте было проведено сравнение лучшего варианта “tng+gti+ng” алгоритма ветвей и отсечений с другим алгоритмом, предложенным в работе Периди и др. (Péridy et al. [180]). По нашей информации данный алгоритм является наилучшим алгоритмом решения задачи $1 \mid r_j \mid \sum w_j U_j$, представленным в литературе. Заметим, что авторы этого алгоритма исследовали его эффективность на компьютере с процессором 450 МГц., поэтому мы установили временно́й лимит в 1000 секунд на время исполнения алгоритма ветвей и отсечений. Сравнение двух алгоритмом можно сделать по таблице 4.2.

Результаты показывают, что предложенный нами алгоритм работает быстрее и решает больший процент примеров в пределах временно́го лимита, даже учитывая разницу в скорости использованных компьютеров.

В таблице 4.3 представлены значения дополнительных параметров, полученных при исследовании варианта “tng+gti+ng”.

Можно увидеть, что в сравнении с вариантом “ng”, среднее число узлов

Таблица 4.2: Сравнение результатов двух алгоритмов

	Алгоритм из [180]			Вариант “tng+gti+ng”		
Тест	P_{1h}	T_{av}	T_{max}	P_{1000s}	T_{av}	T_{max}
b-50	100%	47.1	409.1	100%	1.5	45.8
b-60	100%	157.9	2147.0	100%	3.5	48.5
b-70	97.8%	168.6	1745.3	100%	6.7	117.5
b-80	97.8%	294.6	3567.6	98.9%	12.7	408.2
b-90	88.9%	383.0	3542.7	98.9%	26.3	311.2
b-100	83.3%	515.7	3581.6	98.9%	75.6	930.9
s-40	100%	26.6	448.6	100%	0.6	6.7
s-60	99.4%	178.2	2127.5	100%	3.4	263.8
s-80	95.0%	496.5	3552.2	100%	11.8	514.3
s-100	84.7%	1049.9	3560.0	99.7%	29.6	953.5

Таблица 4.3: Влияние предложенных отсечений

	“ng”		“tng+gti+ng”			
Тест	N_{av}	C_{av}	N_{av}	C_{av}	$PMCE_{av}$	$PMCL$
b-50	396.6	85.6	219.9	10.8	42.9%	0.3%
b-60	768.0	184.5	345.8	13.8	46.7%	2.7%
b-70	1307.1	273.9	527.5	16.2	57.6%	0.1%
b-80	1917.9	252.6	1840.3	85.1	42.1%	0.2%
b-90	2358.6	274.4	1300.7	20.3	51.6%	0.4%
b-100	–	–	3657.9	60.8	51.5%	0.1%
s-40	188.0	30.5	85.3	4.1	34.9%	0.1%
s-60	400.1	80.7	267.5	10.3	42.0%	1.0%
s-80	1019.3	159.5	501.0	16.3	39.1%	2.2%
s-100	–	–	849.8	12.0	48.9%	1.6%
s-120	–	–	1816.1	17.3	44.3%	11.7%
s-140	–	–	2744.3	23.1	42.2%	9.3%

в дереве поиска, необходимое для оптимального решения примеров, значительно меньше. Более того, среднее число необходимых отсечений, построенных по предложенным методам, на порядок меньше, чем число стандартных отсечений “no-good”. Одной из причин, по всей видимости, является эффективность модифицированного алгоритма Карлиера – около 40–50% отсечений в среднем. Хорошим результатом является то, что данный алгоритм работает достаточно быстро, и требуемое число узлов в дереве поиска в подавляющем большинстве случаев не превышает лимит. Сложности возникают только для примеров наибольшей размерности из набора “s”. При их решении лимит достигается около 10% случаев.

Результаты для варианта алгоритма “tng+gti+ng” и набора примеров “d”, где веса зависят от продолжительностей обслуживания, представлены в

таблице 4.4. Как и ожидалось, данные примеры сложнее для решения. И чем сильнее зависимость (меньше параметр δ), тем меньше примеров мы можем решить в пределах лимита времени.

Таблица 4.4: Результаты для набора примеров “d”

Тест	P_{1h}	T_{av}	N_{av}	C_{av}
s-80	100%	11.8	501.0	16.3
s-100	100%	33.0	849.8	12.0
s-120	99.7%	104.9	1816.0	17.3
d-25-80	99.4%	38.7	2055.6	36.6
d-25-100	96.3%	162.8	4728.1	29.3
d-25-120	92.5%	163.0	3486.6	44.9
d-10-80	96.9%	64.6	3675.5	47.6
d-10-100	95.0%	187.2	6045.8	69.8
d-10-120	86.9%	186.2	4327.7	59.0

Для следующего, последнего эксперимента, мы разделили примеры из набора “s” на классы в зависимости от параметра K_2. Чем больше этот параметр генерации, тем больше в среднем запас обслуживания требований $m_j = d_j - r_j - p_j$. По результатам для варианта “tng+gti+ng”, представленным в таблице 4.5 можно сказать, что эффективность алгоритма сильно зависит от типа примеров.

Примеры с небольшими запасами обслуживания решить сложнее. Требуется намного больше узлов в дереве поиска и отсечений для решения таких примеров. Но, с другой стороны, модифицированный алгоритм Карлиера более чем в два раза эффективен при решении этого класса примеров. И, более того, лимит на количество узлов в дереве поиска данного алгоритма, никогда не достигается. То есть мы можем сделать вывод, что, хотя класс примеров с небольшими запасами обслуживания хуже поддаётся решению, но применение предложенных в данной работе отсечений позволяет оптимально решать большой процент таких примеров.

Таблица 4.5: Результаты для классов примеров с различными запасами обслуживания из набора “s”

Тест	P_{1h}	T_{av}	N_{av}	C_{av}	$PMCE_{av}$	$PMCL$
$K_2 = 1$ (наименьший средний запас обслуживания)						
s-140	90.0%	665.0	9323.8	63.6	70.7%	0.0%
s-120	100%	326.4	5784.7	40.5	71.5%	0.0%
s-100	100%	86.0	2372.4	27.4	72.8%	0.0%
s-80	100%	12.9	513.6	12.4	72.1%	0.0%
s-60	100%	2.3	104.5	4.4	69.3%	0.0%
$K_2 = 5$						
s-140	97.5%	77.5	1026.5	21.1	22.8%	23.0%
s-120	98.8%	48.7	842.5	20.6	25.1%	21.2%
s-100	100%	16.0	540.1	11.4	36.3%	2.1%
s-80	100%	17.4	1040.1	37.6	39.3%	0.0%
s-60	100%	3.0	322.1	13.8	36.0%	0.1%
$K_2 = 10$						
s-140	97.5%	18.3	338.8	3.8	25.9%	6.3%
s-120	100%	22.6	414.9	6.2	15.3%	17.6%
s-100	100%	10.4	321.5	8.2	18.4%	3.1%
s-80	100%	7.0	295.1	11.4	19.6%	1.1%
s-60	100%	5.2	467.5	16.9	47.3%	0.6%
$K_2 = 20$ (наибольший средний запас обслуживания)						
s-140	100%	24.0	288.0	3.9	14.4%	5.3%
s-120	100%	21.8	222.4	1.8	15.6%	3.7%
s-100	100%	19.6	165.1	1.1	22.9%	7.6%
s-80	100%	10.1	155.4	3.8	12.0%	24.3%
s-60	100%	3.4	176.1	6.0	27.5%	4.5%

Глава 5

Исследование свойств задачи $1 \,||\, \sum T_j$

В данной главе рассматриваются комбинаторные свойства классической NP—трудной задачи теории расписаний: минимизация суммарного запаздывания для одного прибора $1 \,||\, \sum T_j$. Получены свойства оптимальных расписаний, приводятся алгоритмы решения задачи на основе полученных свойств, а также некоторые оценки оптимального значения целевой функции. Приведены результаты экспериментальных исследований.

В разделе 5.1 приводится постановка исследуемой задачи, вводятся необходимые понятия и обозначения. Раздел 5.2 посвящён декомпозиционным свойствам задачи $1 \,||\, \sum T_j$. В этом разделе приводятся два алгоритма построения оптимальных расписаний для общего случая задачи на основе перебора подходящих позиций для требования с максимальной продолжительностью обслуживания в оптимальном расписании и дальнейшему разбиению примера на подпримеры по подходящим позициям. В разделе 5.3 приводится алгоритм решения примеров, для которых при любом расписании запаздывает постоянное количество требований. В разделе 5.4 строятся метрики для исследуемой задачи. В разделе 5.5. — канонические примеры, в разделе 5.6 — трудоемкости алгоритмов. В разделе 5.7 приводятся некоторые оценки оптимального значения целевой функции, полученные с помощью исследования свойств SPT-расписаний. Раздел 5.8 посвящён результатам экспериментальных исследований.

5.1 Постановка задачи суммарного запаздывания для одного прибора

Необходимо обслужить n требований на одном приборе. Прерывания обслуживания требований, искусственные простои прибора при обслуживании и обслуживание более одного требования в любой момент времени запрещены.

Требования пронумерованы числами $1, 2, \ldots, n$. Множество $N = \{1, 2, \ldots, n\}$ назовем *множеством требований*.

Для требования $j \in N$ заданы следующие *параметры*: *продолжительность обслуживания* $p_j > 0$, $p_j \in \mathbb{Z}^+$, и *директивный срок* окончания обслуживания d_j. Задан *момент начала обслуживания* t_0, с которого прибор готов начать обслуживание требований. Все требования поступают на обслуживание одновременно в момент времени t_0.

Расписание обслуживания требований множества N задаётся кусочно-постоянной непрерывной слева функцией $s : \mathbb{R} \to \{0, 1, 2, \ldots, n\}$. Если $s(t) = 0$, то в момент времени t прибор простаивает; если $s(t) = j$, $j \in N$, то в момент времени t прибор обслуживает требование j. Поскольку в рассматриваемой задаче требования поступают на обслуживание одновременно, обслуживаются на приборе без прерываний и искусственных простоев прибора, то расписание однозначно задаётся перестановкой π элементов множества N. Далее в работе понятие расписания и перестановки множества требований будем отождествлять и называть *расписанием* перестановку π.

Индивидуальный пример исследуемой задачи (далее, *пример*) с заданными множеством требований N, продолжительностями обслуживания p_j, директивными сроками d_j и моментом начала обслуживания t_0 будем обозначать через $I = \langle \{p_j, d_j\}_{j \in N}, t_0 \rangle$. В случае, если параметры требований фиксированы (однозначно определены контекстом), для обозначения примера I будем использовать запись $\{N, t_0\}$. Множество всех $n!$ расписаний для примера I будем обозначать через $\Pi(I)$.

Как и раньше величину $C_j(\pi)$ будем называть *моментом окончания обслуживания* требования j при расписании π. Моменты окончания обслуживания требований при расписании $\pi = (j_1, j_2, \ldots, j_n)$ вычисляются следующим образом:

$$\begin{aligned} C_{j_1}(\pi) &= t_0 + p_{j_1}; \\ C_{j_k}(\pi) &= C_{j_{k-1}}(\pi) + p_{j_k}, \ k = 2, 3, \ldots, n. \end{aligned}$$

Если обслуживание требования i предшествует обслуживанию требования j при расписании π (т.е. выполняется $C_i(\pi) < C_j(\pi)$), то будем использовать запись $(i \to j)_\pi$. Используя это обозначение, момент окончания обслуживания требования $j \in N$ при расписании π можно записать как $C_j(\pi) = t_0 + \sum\limits_{i \in N : (i \to j)_\pi} p_i + p_j$. Обслуживание всех требований примера I завершается в момент времени $t_0 + \sum\limits_{j \in N} p_j$ при любом расписании $\pi \in \Pi(I)$.

Под *запаздыванием* требования $j \in N$ при расписании π будем понимать величину

$$T_j(\pi) = \max\{0, C_j(\pi) - d_j\}.$$

Суммарное запаздывание требований при расписании π определяется как

$$F(\pi) = \sum_{j=1}^{n} T_j(\pi).$$

Задача минимизации суммарного запаздывания заключается в построении оптимального расписания $\pi^* \in \Pi(I)$, *при котором выполняется неравенство* $F(\pi^*) \leq F(\pi)$ *для всех расписаний* $\pi \in \Pi(I)$. Отметим, что данная задача является NP-трудной [114]. Известен псевдополиномиальный алгоритм её решения [150] трудоёмкости $O(n^4 \sum\limits_{j \in N} p_j)$ операций, основанный на методе динамического программирования.

Через $\Pi^*(I)$ будем обозначать множество всех оптимальных расписаний для примера I. Для обозначения величины оптимального суммарного запаздывания примера I будем использовать запись $F^*(I)$.

Покажем, что без ограничения общности рассуждений можно предполагать, что рассматриваются только те примеры I, для которых выполняется

$$d_j \in \left[t_0; t_0 + \sum_{i \in N} p_i \right], \quad \forall\, j \in N. \tag{5.1}$$

В случае если для примера I условия (5.1) не выполняется, построим пример $I' = \left\langle \{p_j, d'_j\}_{j \in N}, t_0 \right\rangle$ такой, что

$$d'_j = \min \left\{ \max\{t_0, d_j\}, t_0 + \sum_{i \in N} p_i \right\}, \quad \forall\, j \in N.$$

Если для некоторого требования $j \in N$ выполняется $d_j > t_0 + \sum\limits_{i \in N} p_i$, то имеем $d'_j = t_0 + \sum\limits_{i \in N} p_i$. Это означает, что требование j не запаздывает при любом расписании π для обоих примеров I и I'. Таким образом, можно исключить такое требование j из рассмотрения и после построения оптимального расписания для редуцированного примера добавить требование j на последнюю позицию в построенное расписание.

Если для некоторого требования $j \in N$ выполняется $d_j < t_0$, то имеем $d'_j = t_0$, и требование j запаздывает при любом расписании π. Лоулером [150] была доказана следующая теорема.

Теорема 5.1 [150] *Пусть* π^* – *оптимальное расписание примера* $I_1 = \langle \{p_j, d_j\}_{j\in N},\ t_0\rangle$. *Выберем величины* d'_j *так, что*

$$\min\{d_j, C_j(\pi^*)\} \le d'_j \le \max\{d_j, C_j(\pi^*)\} \quad \forall j \in N. \tag{5.2}$$

Тогда любое оптимальное расписание примера $I_2 = \left\langle \{p_j, d'_j\}_{j\in N}, t_0 \right\rangle$ *будет оптимальным и для примера* I_1. □

В нашем случае, условия (5.2) для примеров I и I' выполняются, поскольку при любом расписании π справедливо неравенство $t_0 < C_j(\pi) \le t_0 + \sum\limits_{i\in N} p_i$, $j \in N$. Следовательно, любое оптимальное расписание для примера I' будет оптимальным и для примера I. Таким образом, параметры требований примера I' удовлетворяют условиям (5.1), и выполняется $\Pi^*(I') \subseteq \Pi^*(I)$.

Необходимо заметить, что не любое оптимальное расписание примера I будет оптимальным и для примера I'.

Как следствие Теоремы 5.1 может быть сформулировано следующее утверждение.

Лемма 5.1 *Для любого оптимального расписания* π *и любого незапаздывающего требования* j *верно: все требования, которые обслуживаются в интервале* $[C_j(\pi), d_j]$ *не запаздывают.*

□

Оставим доказательство данного утверждения заинтересованному читателю.

Два примера $I = \langle \{p_j, d_j\}_{j\in N}, t_0\rangle$ и $I' = \left\langle \{p'_j, d'_j\}_{j\in N}, t'_0 \right\rangle$ будем называть *равными*, если множества оптимальных расписаний для обоих примеров совпадают, т.е. $\Pi^*(I) = \Pi^*(I')$. Через $T'_j(\pi)$ обозначим значение запаздывания требования j при некотором расписании π, вычисленное для примера I'. Для нашего исследования будут полезны следующие случаи равенства примеров.

(1) Примеры I и I' равны, если $p'_j = p_j$, $d'_j = d_j + C$, $j \in N$, и $t'_0 = t_0 + C$, где C – произвольная константа. Действительно, в этом случае, для любого расписания π и любого требования $j \in N$, обслуживаемого при расписании π, выполняется

$$\begin{aligned} T'_j(\pi) &= \max\left\{0, t_0 + C + \sum_{i:(i\to j)_\pi} p_i + p_j - (d_j + C)\right\} = \\ &= \max\left\{0, t_0 + \sum_{i:(i\to j)_\pi} p_i + p_j - d_j\right\} = \\ &= \max\{0, C_j(\pi) - d_j\} = T_j(\pi). \end{aligned}$$

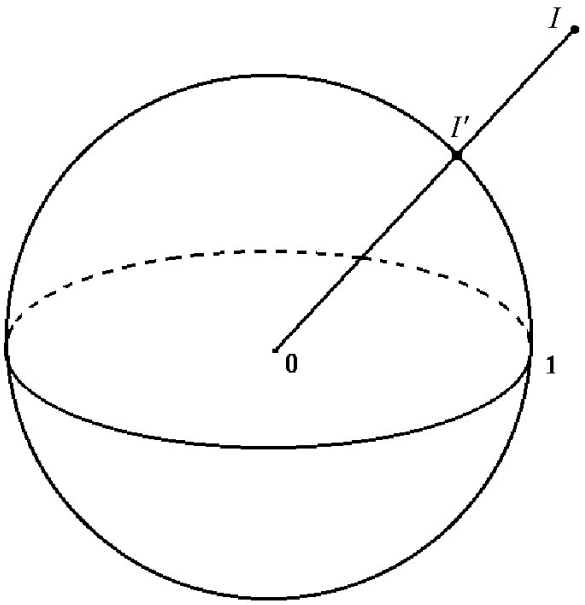

Рис. 5.1: Схематичное изображение проецирования примера I на единичную сферу в $(2n+1)$-мерном евклидовом пространстве. Примеры I и I' равны.

Следовательно $F^*(I) = F^*(I')$ и $\Pi^*(I) = \Pi^*(I')$. В этом смысле, без потери общности, момент начала обслуживания требований можно считать равным нулю, $t_0 = 0$. Однако, в главе 6 нам потребуется выполнение некоторых условий относительно директивных сроков, например $d_n \in \mathbb{Z}^+$, которое будет сделано путём изменения момента начала обслуживания t_0. Поэтому, в работе мы будем предполагать, что t_0 – произвольная величина.

(2) Примеры I и I' равны, если $p'_j = \alpha p_j$, $d'_j = \alpha d_j$, $j \in N$, и $t'_0 = \alpha t_0$, где $\alpha > 0$ – произвольная положительная константа. Действительно, в этом случае, для любого расписания π и любого требования $j \in N$, обслуживаемого при расписании π, выполняется

$$T'_j(\pi) = \max\left\{0, \alpha t_0 + \sum_{i:(i\to j)_\pi} \alpha p_i + \alpha p_j - \alpha d_j\right\} =$$
$$= \alpha \max\left\{0, t_0 + \sum_{i:(i\to j)_\pi} p_i + p_j - d_j\right\} = \alpha T_j(\pi).$$

Следовательно, $F^*(I) = \alpha F^*(I')$ и $\Pi^*(I) = \Pi^*(I')$. Рассмотрим любой

пример I как точку в $(2n+1)$-мерном евклидовом пространстве с координатами $(t_0, p_1, \ldots, p_n, d_1, \ldots, d_n)$. Тогда, все примеры (точки), располо-

женные на луче, исходящем из нуля пространства, равны между собой (рис. 5.1). Поэтому, можно предполагать, что рассматриваются только те примеры (точки), которые лежат на единичной сфере в рассматриваемом пространстве. Однако, в общем случае данный подход неприемлем при построении оптимального расписания некоторым алгоритмом, для работы которого необходимо, чтобы продолжительности обслуживания требований были целочисленными величинами. Поэтому ограничимся выбором константы $\alpha = 1/\mathbf{НОД}\,(p_1, \ldots, p_n)$.

Необходимо также отметить, что для любых двух равных примеров I и I' выполняется свойство: если требование при оптимальном расписании π примера I запаздывает (не запаздывает), то и в примере I' при расписании π требование запаздывает (не запаздывает). Доказательство этого можно провести от противного, что и предлагаем сделать заинтересованному читателю.

Введём некоторые дополнительные обозначения и понятия.

Расписание π будем называть SPT[1] — расписанием и обозначать через π_{spt}, если требования упорядочены при данном расписании в порядке неубывания продолжительностей обслуживания, т.е. для любых двух требований i и j таких, что $p_i < p_j$, выполняется $(i \to j)_{\pi_{spt}}$.

Расписание π будем называть EDD[2] — расписанием и обозначать через π_{edd}, если требования упорядочены при данном расписании в порядке неубывания директивных сроков, т.е. для любых двух требований i и j таких, что $d_i < d_j$, выполняется $(i \to j)_{\pi_{edd}}$.

Через $\{\pi\}$ будем обозначать множество требований, упорядоченных при расписании π. В случае $\{\pi\} \neq N$ ($\{\pi\} \subset N$) расписание π будем называть *частичным* расписанием.

Запись $(i \to j)_\pi$ при необходимости будет расширена до $(i \to j \to k)_\pi$ для обозначения порядка обслуживания более, чем двух требований при некотором расписании π. Также, будем использовать запись $(j \to N')_\pi$ или $(N' \to N'')_\pi$ для обозначения порядка обслуживания между некоторыми множествами требований.

Для обозначения структуры расписания π будем использовать запись $\pi = (\pi_1, \pi_2, \ldots, \pi_m)$, где расписание π_i, $i = 1, \ldots, m$, является частичным расписанием (подрасписанием расписания π). При этом, для частичных расписаний π_i выполняется $\{\pi\} = \bigcup\limits_{i=1}^{m} \{\pi_i\}$, $\{\pi_i\} \bigcap \{\pi_j\} = \emptyset$ для $i \neq j$, $(\{\pi_1\} \to \{\pi_2\} \to$

[1] Shortest Processing Time.
[2] Earliest Due Date.

$\ldots \to \{\pi_m\})_\pi$, и требования каждого частичного расписания π_i, $i = 1, \ldots, m$, упорядочены в том же порядке, что и при расписании π. Для обозначения структуры расписания π относительно некоторого требования $j \in \{\pi\}$ будем использовать запись $\pi = (\pi_1, j, \pi_2)$.

Пользуясь нотацией Грэхема [125], для обозначения исследуемой задачи будем использовать запись $1 \mid\mid \sum T_j$. Рассматриваемая задача $1 \mid\mid \sum T_j$ является NP-трудной в обычном смысле [114].

В заключение данного раздела приведём пример связи рассматриваемой задачи для одного прибора $1 \mid\mid \sum T_j$ с некоторой задачей оптимального упорядочения набора требований, возникающей на практике. Рассмотрим некоторый обслуживающий центр (ОЦ), процесс работы которого разбит на циклы, состоящие из двух этапов. На первом этапе центр набирает заявки и на втором этапе осуществляет их исполнение. В ОЦ имеется m однородных исполнителей, каждая заявка может быть исполнена любым из них. Исполнители обладают, в общем случае, различной производительностью и выполняют в любой момент времени не более одной заявки. Выполнение заявки не может быть прервано, т.е. если исполнитель начинает выполнение некоторой заявки, то он выполняет её до конца. При приёме заявки ОЦ и клиент договариваются о сумме, которую ОЦ получит за выполнение заявки, а также о моменте времени, к которому заявка должна быть исполнена. При выполнении некоторой заявки позднее обговоренного момента времени ОЦ выплачивает клиенту фиксированный для любой заявки штраф за каждую единицу времени, прошедшую после этого момента. Целью составления расписания работы ОЦ является минимизация суммы штрафных выплат по всем принятым к исполнению заявкам.

Данная задача может быть сформулирована в терминах теории расписаний следующим образом. Задано множество требований $N = \{1, 2, \ldots, n\}$ и множество параллельных приборов $M = \{1, 2, \ldots, m\}$. Каждое требование $j \in N$ имеет директивный срок d_j, к которому желательно завершить обслуживание данного требования, и продолжительность p_{ji} обслуживания требования j на приборе $i \in M$. Требования обслуживаются на приборах без прерываний, в любой момент времени любой прибор обслуживает не более одного требования. Расписание обслуживания требований π строится с момента времени $t_0 = 0$ и для данной задачи может быть задано в виде набора из m перестановок $\pi_1, \pi_2, \ldots, \pi_m$, где π_i, $i \in M$, задаёт порядок обслуживания требований множества $\{\pi_i\}$ на приборе i. При этом, $\{\pi_1\}, \{\pi_2\}, \ldots, \{\pi_m\}$ задаёт разбиение множества требований N (т.е. $\bigcup\limits_{i \in M} \{\pi_i\} = N$ и $\{\pi_i\} \bigcap \{\pi_j\} = \emptyset$ для $i, j \in M$, $i \neq j$). Момент окончания $C_j(\pi)$ требования $j \in N$ при распи-

сании π вычисляется описанным выше образом. Требуется построить такое расписание обслуживания требований множества N на приборах из множества M, при котором минимизируется суммарное запаздывание требований, т.е. $\sum\limits_{j\in N} \max\{0, C_j(\pi) - d_j\} \ \rightarrow \ \min$.

Имея некоторый алгоритм, основанный на идее метода ветвей и границ, для задачи суммарного запаздывания для параллельных приборов было бы желательно иметь некоторый способ вычисления нижних оценок оптимального значения целевой функции. Это может быть сделано с помощью использования методов решения и построения оценок для примеров задачи $1 \,||\, \sum T_j$. С другой стороны, алгоритм решения задачи суммарного запаздывания для параллельных приборов может быть организован так, что сначала строится разбиение множества требований N на m подмножеств, которые будут обслуживаться на соответствующих приборах, а затем решается m независимых примеров задачи $1 \,||\, \sum T_j$. В этом случае, исходная задача для многих приборов допускает разбиение на несколько независимых подзадач для одного прибора.

5.2 Алгоритмы построения оптимальных расписаний, основанные на "декомпозиционных" свойствах задачи

Перенумеруем требования множества N примера $I = \{N, t_0\}$ в порядке неубывания директивных сроков:

$$d_1 \leq d_2 \leq \ldots \leq d_n,$$

если $d_j = d_{j+1}$, то $p_j \leq p_{j+1}$, $j = 1, 2, \ldots, n-1$.

Среди требований множества N выделим требование j^* такое, что

$$j^* = \max\left\{ j \in N : p_j = \max_{i\in N} p_i \right\}.$$

Требование j^* является требованием с максимальной продолжительностью обслуживания среди требований множества N; если таких требований в множестве N несколько, то выбирается требование с наибольшим номером.

Пусть $S_k = t_0 + \sum\limits_{j=1}^{k} p_j$, $k = 1, 2, \ldots, n$. Величина S_k равна моменту окончания обслуживания требования k при EDD расписании $(1, 2, \ldots, n)$.

Согласно выбранной нумерации требований множества N и определения требования j^* выполняются следующие условия:

$$\begin{cases} p_{j^*} > p_j, d_{j^*} \leq d_j, \text{ для всех } j \in \{j^*+1, \ldots, n\}; \\ p_{j^*} \geq p_j, d_{j^*} \geq d_j, \text{ для всех } j \in \{1, 2, \ldots, j^*-1\}. \end{cases} \tag{5.3}$$

Определим множество $L(I)$ *подходящих позиций* для требования j^* и покажем, что существует оптимальное расписание $\pi^* \in \Pi^*(I)$, при котором требование j^* обслуживается на некоторой позиции из множества $L(I)$. При этом, известны множества требований, обслуживание которых предшествует и следует за обслуживанием требования j^*.

Определение 5.1 [24] *множества подходящих позиций для требования j^*. Определим через $L(I)$ множество всех индексов $k \geq j^*$ таких, что*

(a) $d_j + p_j \leq S_k$ для всех $j \in \{j^*+1, \ldots, k\}$;

(b) $S_k \leq d_{k+1}$.

доопределив $d_{n+1} := +\infty$. Множество $L(I)$ будем называть множеством подходящих позиций для требования j^*.

Для нахождения множества подходящих позиций $L(I)$ для требования j^* необходимо выполнить $O(n)$ операций.

Теорема 5.2 *Для любого примера I множество подходящих позиций $L(I)$ для требования j^* не пусто.*

Доказательство. Предположим, что $L(I) = \emptyset$. То есть ни одна из позиций $k \geq j^*$ не является подходящей. Позиция $k = j^*$ не является подходящей тогда и только тогда, когда выполняется $S_{j^*} > d_{j^*+1}$.

Позиция $k = n$ не является подходящей тогда и только тогда, когда существует такое требование $j_0 \in \{j^*+1, \ldots, n\}$, что $d_{j_0} + p_{j_0} > S_n$.

Рассмотрим случай, когда $j_0 = j^* + 1$. Поскольку для требования j_0 выполняется $d_{j_0} + p_{j_0} > S_n = S_{j^*} + p_{j_0} + p_{j_0+1} + \ldots + p_n$ и $p_j > 0$, $\forall\, j \in N$, то справедливо неравенство $d_{j_0} > S_{j^*}$. Учитывая $j_0 = j^* + 1$, получаем, что позиция $k = j^*$ является подходящей, что противоречит предположению $L(I) = \emptyset$.

Пусть требование $j_0 \in \{j^*+2, \ldots, n\}$. При этом, выполняется

$$d_{j_0} + p_{j_0} > S_n = S_{j_0-1} + p_{j_0} + p_{j_0+1} + \ldots + p_n.$$

Поскольку $p_j > 0 \quad \forall j \in N$, то справедливо неравенство $d_{j_0} > S_{j_0-1}$. Согласно предположению $L(I) = \emptyset$ получаем, что для того, чтобы позиция $k = j_0 - 1$ не была подходящей необходимо, чтобы выполнилось одно из неравенств:

$$\begin{gathered} d_{j^*+1} + p_{j^*+1} > S_{j_0-1}; \\ d_{j^*+2} + p_{j^*+2} > S_{j_0-1}; \\ \dots \quad \dots \\ d_{j_0-1} + p_{j_0-1} > S_{j_0-1}. \end{gathered}$$

Пусть для некоторого $i \in \{1, \dots, j_0 - j^* - 1\}$ выполняется неравенство $d_{j^*+i} + p_{j^*+i} > S_{j_0-1}$. В этом случае, мы имеем

$$d_{j^*+i} + p_{j^*+i} > S_{j_0-1} \geq S_{j^*+i}.$$

Следовательно, выполняется $d_{j^*+i} > S_{j^*+i-1}$. Далее, чтобы позиция $k = j^* + i - 1$ не была подходящей, необходимо, чтобы выполнилось одно из следующих неравенств:

$$\begin{gathered} d_{j^*+1} + p_{j^*+1} > S_{j^*+i-1}; \\ d_{j^*+2} + p_{j^*+2} > S_{j^*+i-1}; \\ \dots \quad \dots \\ d_{j^*+i-1} + p_{j^*+i-1} > S_{j^*+i-1}. \end{gathered}$$

Продолжая таким образом, мы получим, что согласно предположению $L(I) = \emptyset$, должно выполнится условие $d_{j^*+1} + p_{j^*+1} > S_{j^*+1}$. Из последнего неравенства получаем $d_{j^*+1} > S_{j^*}$, это означает, что позиция $k = j^*$ является подходящей. Мы получили противоречие с исходным предположением, что $L(I) = \emptyset$. Теорема доказана. □

Следствие 5.1 [24] *Существует оптимальное расписание $\pi^* \in \Pi^*(I)$, при котором требование j^* обслуживается на позиции $k \in L(I)$. При этом, для подходящей позиции k выполняется:*

$$\begin{cases} (j \to j^*)_{\pi^*} \text{ для всех } j \in \{1, 2, \dots, k\} \setminus \{j^*\} \text{ и} \\ (j^* \to j)_{\pi^*} \text{ для всех } j \in \{k+1, \dots, n\}. \end{cases} \tag{5.4}$$

□

Доказательство. То что существует оптимальное расписание, при котором выполняется (5.4), следует из теоремы, доказанной Лоулером (Lawler E.L., [150]) в 1977 году. □

Согласно теореме 5.2, если не существует такого требования j_0, что $d_{j_0} + p_{j_0} > S_n$, то позиция $k = n$ является подходящей для требования j^*. В противном случае, если существует такое требование j_0, тогда существует такое k, $j^* \le k \le j_0$, что $k \in L(I)$.

Лемма 5.2 [144, 186, 21] *Для любого примера* $\langle \{p_j, d_j\}_{j\in N}, t_0\rangle$ *существует оптимальное расписание* π^* *обслуживания требований множества* N, *при котором* $(j \to j^*)_{\pi^*}$ *для всех требований* $j \in \{1, 2, \ldots, k\} \setminus \{j^*\}$ *и* $(j^* \to j)_{\pi^*}$ *для всех требований* $j \in \{k+1, \ldots, n\}$ *для некоторого* $k \in L(N, t_0)$. □

Рассмотрим пример (подпример) обслуживания требований множества $N' \subseteq N$, $N' = \{1, 2, \ldots, n'\}$, с момента времени $t' \ge t_0$. Множество $L(N', t')$ есть множество всех индексов $k \in \{1, \ldots, n'\}$, $k \ge j^*(N')$, таких что:

(а) $t' + \sum_{j=1}^{k} p_j < d_{k+1}$ (**правило исключения 1** [24, 208]) и

(б) $d_j + p_j \le t' + \sum_{j=1}^{k} p_j$, для всех $j = \overline{j^*(N')+1, k}$ (**правила исключения 2, 3** [24, 208]),

где $d_{n'+1} := +\infty$.

Опишем алгоритм нахождения оптимального расписания на основе использования правил исключения 1 – 3.

Алгоритм 5.1 A

1: π^*:=**ProcL**(N, t_0).

Алгоритм 5.2 Процедура **ProcL**(N, t)

1: Дан пример $\{N, t\}$, множество требований $N = \{j_1, j_2, \ldots, j_n\}$ и момент начала обслуживания t, $d_{j_1} \le d_{j_2} \le \ldots \le d_{j_n}$;
2: **if** $N = \emptyset$ **then**
3: $\pi^* := \emptyset$, $RETURN$ π^*;
4: **end if**
5: Найдём требование $j^*(N, t)$ из множества N с момента времени t;
6: Найдём множество $L(N, t)$ для требования j^*;
7: **for all** $k \in L(N, t)$ **do**
8: $\pi_k := (\mathbf{ProcL}(N', t'), j^*, \mathbf{ProcL}(N'', t''))$, где
$N' := \{j_1, \ldots, j_k\} \setminus \{j^*\}$, $t' := t$,
$N'' := \{j_{k+1}, \ldots, j_n\}$, $t'' := t + \sum_{i:=1}^{k} p_{j_i}$;
9: **end for**
10: π^*:=$\arg \min_{k\in L(N,t)} \{F(\pi_k, t)\}$; $RETURN$ π^*.

Пусть $N = \{j_1, \ldots, j_n\}$, $d_{j_1} \leq \ldots \leq d_{j_n}$. Модифицированное EDD-расписание, при котором требование $j^* = j_m$, $m \leq k$, обслуживается k-м по порядку, обозначим через $\pi^k = (j_1, \ldots, j_{m-1}, j_{m+1}, \ldots, j_k, j^*, j_{k+1}, \ldots, j_n)$.

Лемма 5.3 Правило исключения 4 [105, 208]. *Если $F(\pi^k) > F(\pi^{k+1})$ или $F(\pi^k) \geq F(\pi^i)$ для требования $j^* \leq i < k$, то позиция k исключается из списка "подходящих" позиций $L(N', t')$ для задачи $\langle \{p_j, d_j\}_{j \in N'}, t' \rangle$, если для множества $|L(N', t')| > 1$.* □

Пусть B_j – множество требований, обслуживающихся перед требованием j, а A_j – множество требований обслуживающихся после требования j при любом оптимальном расписании. Множества B_j и A_j могут быть пустыми одновременно.

Определим $E_j = P(B_j) + p_j + t'$, $L_j = P(N' \setminus A_j) + t'$ как наименьшее и наибольшее значения момента окончания C_j обслуживания требования j при любом оптимальном расписании обслуживания требований множества N' с момента времени t'.

Лемма 5.4 Условия Эммонса [117]. *Существует оптимальное расписание π^*, при котором*

1) i предшествует j, $(i \to j)_{\pi^*}$, если $d_i \leq \max(E_j, d_j)$ и $p_i \leq p_j$;

2) j предшествует i, $(j \to i)_{\pi^*}$, если для этих требований выполняется $d_i + p_i \geq L_j$ и $d_i > \max(E_j, d_j)$, $p_i \leq p_j$. □

Согласно условиям Эммонса существует оптимальное расписание $\pi^* \in \Pi^*(I)$, при котором выполняется

$$(j \to j^*)_{\pi^*}, \text{ для всех } 1 \leq j < j^*. \tag{5.5}$$

Согласно свойству оптимальных расписаний, которое утверждается в следствие 5.1, решение любого примера рассматриваемой задачи сводится к решению нескольких независимых друг от друга подпримеров. Поэтому это свойство называют *декомпозиционным свойством* задачи.

Далее приводятся два алгоритма решения примеров задачи $1 \,||\, \sum T_j$ с использованием декомпозиционного свойства. Алгоритмы основаны на разбиении (декомпозиции) исходного примера задачи на подпримеры по подходящим позициям для требования с максимальной продолжительностью обслуживания. Отметим, что для выполнения приводимых алгоритмов не требуется условие целочисленности продолжительностей обслуживания требований. Опишем основную идею данных алгоритмов.

Рассмотрим некоторый пример $I = \{N, t_0\}$. Без ограничения общности предположим, что требования множества N примера $I = \{N, t_0\}$ пронумерованы в порядке невозрастания директивных сроков:

$$d_1 \leq d_2 \leq \ldots \leq d_n, \tag{5.6}$$

при этом, если $d_j = d_{j+1}$, то $p_j \leq p_{j+1}$, $j = 2, 3, \ldots, n$.

Для данного примера I найдём требование j^*, для которого построим множество подходящих позиций $L(I)$. Пусть $m = |L(I)|$. Разобъём пример I на $2m$ подпримеров следующим образом. Для каждого $k \in L(I)$ построим два подпримера $I'_k = \{N'_k, t'_k\}$ и $I''_k = \{N''_k, t''_k\}$, где $N'_k = \{1, 2, \ldots, k\} \setminus \{j^*\}$, $t'_k = t_0$ и $N''_k = \{k+1, \ldots, n\}$, $t''_k = S_k$. Решим построенные примеры, т.е. построим оптимальные расписания π'_k и π''_k для примеров I'_k и I''_k $k \in L(I)$. Пусть $\pi_k = (\pi'_k, j^*, \pi''_k)$. Тогда расписание π_{k^*}, где $k^* = \arg\min\limits_{k \in L(I)} F(\pi_k)$, будет оптимальным расписанием для примера I. Построение примеров I'_k и I''_k для некоторого $k \in L(I)$ будем называть *декомпозицией* примера I по подходящей позиции k. Подпримеры I'_k и I''_k будем называть соответственно "левым" и "правым" подпримерами для позиции k.

Определим *дерево подпримеров* следующим образом. В корне дерева подпримеров находится исходный пример $I = \{N, t_0\}$. На первом уровне дерева находятся подпримеры, полученные с помощью декомпозиции примера I по подходящим позициям из множества $L(I)$. На втором уровне находятся подпримеры, полученные с помощью декомпозиции подпримеров первого уровня по всем подходящим позициям и так далее. В листьях дерева находятся подпримеры, множество требований которых состоит из одного требования.

Идея первого алгоритма (алгоритма A) основана на обходе дерева подпримеров "в глубину", используя рекурсивный вызов процедуры декомпозиции примеров на подримеры (Процедура **ProcL**). Данная процедура принимает в качестве параметров множество требований и момент начала обслуживания некоторого подпримера и результатом её работы является оптимальное расписания для этого примера (подпримера).

В ходе работы алгоритма A процесс разбиения примеров на подпримеры выполняется по всем подходящим позициям для требования с текущей максимальной продолжительностью обслуживания. Следовательно алгоритм A находит оптимальное расписание для любого примера общего случая рассматриваемой задачи.

Отметим следующий недостаток алгоритма A. Каждый подпример $I' = \{N', t'\}$ исходного примера I идентифицируется множеством требований N'

и моментом начала обслуживания t'. Если в дереве подпримеров встретятся два подпримера с одинаковыми множеством требований и моментом начала обслуживания, то в процессе работы алгоритма A дальнейшая декомпозиция этих подпримеров будет осуществлена в обоих случаях.

Для иллюстрации данного момента рассмотрим пример I, для которого выполняется $p_1 > p_2 > \ldots > p_n$, $d_1 < d_2 < \ldots < d_n$ и $d_n - d_1 \leq p_n$. Для данного примера I и всех его подпримеров множество подходящих позиций для требования с максимальной продолжительностью обслуживания j^* состоит не более, чем из двух позиций – первой (требование j^* обслуживается до всех остальных требований) и последней (требование j^* обслуживается после всех остальных требований). Таким образом $L(I) = \{1, n\}$. При декомпозиции примера I по первой позиции получим "правый" подпример I' с множеством требований $N' = \{2, \ldots, n\}$ и моментом начала обслуживания $t' = t_0 + p_1$ (множество требований "левого" подпримера будет пустым). При декомпозиции примера I по последней позиции получим "левый" подпример I'' с тем же множеством требований N' и моментом начала обслуживания $t' = t_0$. Далее, пусть множества подходящих позиций $L(I')$ и $L(I'')$ также состоят из двух позиций. Тогда "левый" подпример примера I'' совпадает с "правым" подпримером примера I'. Алгоритм вынужден будет решать один и тот же подпример дважды.

При рекурсивной реализации алгоритма A происходит просмотр дерева подпримеров в глубину. Каждый подпример I' идентифицируется набором требований N' и моментом начала обслуживания требований t'. Если в процессе работы алгоритма A в разных ветвях дерева будут построены одинаковые подпримеры, то их декомпозиция и дальнейшее решение будет произведено в обоих случаях.

Отметим одно важное свойство алгоритма A.

Будем называть расписание π обслуживания требований множества N с момента времени t локально оптимальным, если перестановка любых двух требований в этом расписании не уменьшает (не улучшает) значения целевой функции.

Все подходящие расписания, которые строятся в процессе работы алгоритма A, являются локально оптимальными. Для данного класса примеров ($p_1 > p_2 > \ldots > p_n$, $d_1 < d_2 < \ldots < d_n$ и $d_n - d_1 \leq p_n$) множество локально оптимальных расписаний будет содержать $O(2^{\frac{n}{2}})$ расписаний

5.2.1 Алгоритм SBA

Для того чтобы избежать повторной декомпозиции одинаковых подпримеров, предлагается использовать алгоритм SBA, который является модификацией алгоритма A. Процесс выполнения алгоритма SBA разбивается на два этапа. На первом этапе осуществляется декомпозиция исходного примера в список всех его подпримеров вплоть до получения подпримеров из одного требования. При построении данного списка подпример включается в список только в том случае, если в данном списке нет их аналога (т.е. в списке нет подпримера, множество требований и момент начала обслуживания которого те же, что и у добавляемого подпримера). На втором шаге осуществляется построение оптимального расписания.

Обозначим через LS список подпримеров примера I, количество элементов в списке LS равно количеству вершин в дереве подпримеров для примера I. Для определенности будем обозначать исходный пример I через I_0 и оптимальное расписание для подпримера $I' = \{N', t'\}$ – через $\pi^*(I')$.

Приведём формальное описание вспомогательной процедуры **ProcD** (алгоритм 5.3) , которая выполняет разбиение некоторого примера на подпримеры по подходящим позициям для требования с максимальной продолжительностью j^* и возвращает список полученных подпримеров RLS.

Алгоритм 5.3 Процедура $ProcD(I)$

1: $RLS := \emptyset$;
2: **if** $N \neq \emptyset$ **then**
3: найдём множество $L(I)$ подходящих позиций для требования j^*;
4: **for all** $k \in L(I)$ **do**
5: построим два подпримера $I'_k = \{N', t'\}$ и $I''_k = \{N'', t''\}$, где
$N' := \{1, 2, \ldots, k\} \setminus \{j^*\}$, $t' := t_0$,
$N'' := \{k+1, \ldots, n\}$, $t'' := S_k$;
6: $RLS := RLS \bigcup \{I'_k, I''_k\}$;
7: **end for**
8: **end if**
9: $RETURN\ RLS$;

Перейдем к алгоритму решения исходной проблемы $1||\sum T_j$ без повторного решения равных подпримеров. Отметим, что строка 5 первого этапа алгоритма SBA (алгоритм 5.3) означает добавление подпримеров в список LS только в том случае, если в списке LS нет аналогов добавляемых подпримеров.

Список подпримеров LS может быть организован как хэш-таблица. Значение хэш-функции, соответствующее подпримеру, вычисляется как зна-

чение некоторой функции от момента начала обслуживания требований данного подпримера. Для ускорения работы процедур поиска подпримеров на втором этапе алгоритма SBA для каждого примера в списке подпримеров предлагается сохранять указатели (ссылки) на все "левые" и "правые" подпримеры данного примера, а также предлагается вести вспомогательную таблицу, в каждой строке m, $m = 1, \ldots, n$ в которой сохраняются указатели (ссылки) на все подпримеры, множество требований которых состоит из m требований.

Алгоритм 5.4 SBA

ЭТАП 1: ПОСТРОЕНИЕ СПИСКА ПОДПРИМЕРОВ
$LS := \{I_0\}$;
for all $I' \in LS$, где подпример I' не помечен **do**
  пометить подпример I'; $LS := LS \bigcup \{\mathbf{ProcD}(I') \setminus \{LS \bigcap \mathbf{ProcD}(I')\}\}$;
end for
ЭТАП 2: ПОСТРОЕНИЕ ОПТИМАЛЬНОГО РАСПИСАНИЯ
for all $I' \in LS$, $I' = \{N', t'\}$ и $|N'| = 1$ **do**
  $\pi^*(I') := \{j\}$, где $N' = \{j\}$;
end for
for $m = 2, 3, \ldots, n$ **do**
  for all $I' \in LS$, где множество требований подпримера I' содержит m требований **do**
    for all $k \in L(I')$ **do**
      найти в списке LS подпримеры I'_k и I''_k, которые были получены на Этапе 1 при декомпозиции примера I' по подходящей позиции k;
      $\pi_k := (\pi^*(I'_k), j^*, \pi^*(I''_k))$;
    end for
    $\pi^*(I') := \pi_{k^*}$, где $k^* = \arg \min\limits_{k \in L(I')} \{F(\pi_k)\}$;
  end for
end for
RETURN $\pi^*(I_0)$.

В таблице 5.1 представлены экспериментальные результаты сравнительного анализа количества вершин в дереве ветвлений алгоритма A и количества элементов в списке подпримеров алгоритма SBA, вычисленные на одних и тех же тестовых примерах. Тестовые примеры генерировались следующим образом. Множество требований тестового примера состоит из $n = 2\bar{n}$ требований, $\bar{n} = 3, 4, \ldots, 13$, момент начала обслуживания равен 0. Продолжительности обслуживания берутся $p_j = 2 + \frac{1}{j^2}$, $j = 1, \ldots, \bar{n}$. Пусть $D = \sum\limits_{j=1}^{\bar{n}} p_j$ и $D' = \sum\limits_{j=\bar{n}}^{2\bar{n}} p_j$. При этом, поскольку $\sum\limits_{j=1}^{\infty} \frac{1}{j^2} = \frac{\pi^2}{6} < 2$, где $\pi = 3.14\ldots$, выполняется $D < D'$. Зададим $d_1 = D$ и $d_{2\bar{n}} = D'$. Выберем $d_2, d_3, \ldots, d_{2\bar{n}-1}$ так, чтобы

Таблица 5.1: Сравнительные результаты количества вершин в дереве подпримеров алгоритма A (H_A) и количества элементов в списке подпримеров алгоритма SBA (H_{SBA}).

n	H_A	H_{SBA}		n	H_A	H_{SBA}
6	34	34		18	92 377	56 668
8	125	125		20	352 715	122 010
10	461	461		22	1 352 077	215 806
12	1 715	1 695		24	5 200 299	337 013
14	6 434	6 165		26	20 058 229	484 843
16	24 309	20 234				

$d_1 < d_2 < \ldots < d_{2\bar{n}-1} < d_{2\bar{n}}$. Отметим, что для данных примеров множество локально оптимальных расписаний содержат $O(2^{\frac{n}{2}})$ расписаний.

В таблице через H_A обозначается количество вершин в дереве подпримеров при решении тестового примера с помощью алгоритма A, H_{SBA} – количество элементов в списке подпримеров при решении тестовых примеров с помощью алгоритма SBA.

Обратим внимание, что с ростом n количество "одинаковых" подпримеров растёт. Например, для $n = 26$ дерево поиска содержало 20 058 229 вершин, из которых только 484 843 вершины уникальны, т.е. меньше 25% от общего количества вершин. Можно сказать, что более 75% времени алгоритм A решал одни и те же примеры по несколько раз...

5.3 Построение оптимальных расписаний в случае фиксированного количества запаздывающих требований

Через $D(\pi)$ будем обозначать множество запаздывающих требований при расписании π, т.е. $D(\pi) = \{j \in \{\pi\} : C_j(\pi) > d_j\}$. В данном разделе рассматриваются свойства оптимальных расписаний и предлагается алгоритм решения примеров I в случае, если при любом расписании $\pi \in \Pi(I)$ выполнены следующие условия:

(i) $|D(\pi)| = m$, где m – некоторая константа, $0 \leq m \leq n$;

(ii) требования множества $D(\pi)$ упорядочены в конце расписания π, т.е. расписание π можно представить в виде $\pi = (\pi_1, \pi_2)$, где $\{\pi_2\} = D(\pi)$.

Сформулируем и докажем некоторые свойства оптимальных расписаний в рассматриваемом случае. Будем обозначать рассматриваемый случай

исследуемой NP–трудной задачи через (F).

Лемма 5.5 *В случае (F) при любом оптимальном расписании π^* требования множества $D(\pi^*)$ обслуживаются в SPT-порядке.*

Доказательство. Предположим противное, т.е. при некотором оптимальном расписании π^* существует пара соседних требований $i, j \in D(\pi^*)$, $(i \to j)_{\pi^*}$, для которых выполняется $p_i > p_j$. Построим расписание π, которое отличается от π^* тем, что требования i и j переставлены местами. Отметим, что при расписании π требования i и j также запаздывают.

При расписаниях π^* и π выполняется

$$F(\pi) - F(\pi^*) = T_j(\pi) - T_j(\pi^*) + T_i(\pi) - T_i(\pi^*).$$

Поскольку i и j — пара соседних запаздывающих требований и количество запаздывающих требований при любом расписании является константой, то $F(\pi) - F(\pi^*) = p_j - p_i < 0$. Это противоречит тому, что π^* является оптимальным расписанием. Лемма доказана. □

Таким образом задача построения оптимального расписания π^* в рассматриваемом случае сводится к нахождению множества запаздывающих требований $D(\pi^*)$, состоящего из m требований. При этом, требования множества $N \setminus D(\pi^*)$ обслуживаются в любом порядке в начале расписания π^*, а затем обслуживаются требования множества $D(\pi^*)$ в порядке неубывания продолжительностей обслуживания. То есть мы имеем аналог задачи РАЗБИЕНИЕ, которую рассмотрим более подробно позже.

Далее, пусть требования примера I пронумерованы в порядке невозрастания продолжительностей обслуживания $p_1 \geq p_2 \geq \ldots \geq p_n$, при этом, если $p_j = p_{j+1}$, $j = 1, \ldots, n-1$, то $d_j \geq d_{j+1}$.

Лемма 5.6 *В случае (F) существует оптимальное расписание π^* такое, что если для некоторого $k \in N$ выполняется $k \in D(\pi^*)$, то $(j \to k)_{\pi^*}$ для всех требований $j \in \{k+1, \ldots, n\}$.*

Доказательство. Так как $j \in \{k+1, \ldots, n\}$, то согласно выбранной нумерации требований выполняется $p_k \geq p_j$. Для доказательства леммы рассмотрим следующие возможные случаи:

а) пусть $p_k > p_j$. Тогда, если $j \in D(\pi^*)$, то согласно лемме 5.5 обслуживание требования j предшествует обслуживанию требования k. Если же $j \notin D(\pi^*)$, то обслуживание требования j предшествует обслуживанию требования k согласно условиям в рассматриваемом случае (условие (ii));

b) пусть $p_k = p_j$. В этом случае выполняется $d_k \geq d_j$. Тогда, согласно условию Эммонса[3], существует оптимальное расписание, при котором обслуживание требования j предшествует обслуживанию требования k.

Лемма доказана. □

Через $j_1, j_2, \ldots, j_m$ будем обозначать требования, упорядоченные на последних m позициях при некотором оптимальном расписании. При этом, согласно лемме 5.6, будем искать оптимальное расписание, при котором $j_1 \in \{m, m+1, \ldots, n\}$, $j_2 \in \{m-1, m, \ldots, n\}$, $\ldots$, $j_n \in \{1, \ldots, n\}$. Действительно, согласно лемме 5.6, все расписания, при которых $j_1 \in \{1, \ldots, m-1\}$ можно исключить из рассмотрения, т.к. при данных расписаниях будет существовать такое k, $1 < k \leq m$, что $j_k \geq m$, т.е. если $j_1 < m$, то на оставшиеся $m-1$ место $j_2, \ldots, j_m$ будут претендовать требования множества $\{1, \ldots, j_1 - 1\}$, количество которых меньше, чем количество требуемых позиций. По той же причине, если $j_k = j$, $1 \leq k \leq m$, то можно исключить из рассмотрения все расписания, при которых $j_i \in \{j+1, \ldots, n\}$, $k < i \leq m$.

Построение оптимального расписания в рассматриваемом случае реализуем в виде левостороннего обхода дерева следующего вида. На k-ом уровне дерева, $1 \leq k \leq m$, расположены вершины, соответствующие требованиям, каждое из которых мы пытаемся поставить на обслуживание k-ым запаздывающим по порядку при оптимальном расписании. Каждой вершине k-го уровня дерева сопоставляется величина, равная наименьшему значению суммарного запаздывания требований, стоящих на "запаздывающих" позициях расписания $k+1, \ldots, m$ плюс запаздывание требования, которое планируется в данной вершине на место k. Все вершины k-го уровня, являющиеся потомками некоторой вершины уровня $(k-1)$, упорядочены в дереве слева направо в порядке возрастания номеров требований, сопоставленных данным вершинам.

На первое запаздывающее место j_1 при некотором оптимальном расписании претендуют требования $m, m+1, \ldots, n$. Таким образом, количество вершин на первом уровне дерева равно $n - m + 1$. Поставим на первое запаздывающее место требование m. Согласно лемме 5.6, на второе место j_2 претендует только требование $m-1$, на третье место j_3 претендует только требование $m-2$ и так далее. На последнее место претендует только требование 1. Таким образом, поддерево с корнем в самой левой вершине первого уровня содержит только одну ветвь с m вершинами, обход которой дает нам порядок обслуживания требований $(m, m-1, \ldots, 1)$.

[3]Напомним, что Эммонсом было установлено существование оптимального расписания π^* такого, что если для требований $i, j \in N$ выполняется $p_i \leq p_j$ и $d_i \leq d_j$, то обслуживание требования i предшествует обслуживанию требования j при оптимальном расписании π^*.

При этом, листовой вершине данной ветви сопоставляется значение $S - d_1$, где $S = t_0 + \sum\limits_{j \in N} p_j$; вершине уровня $(m-1)$ — значение $(S - d_1) +$ $+ (S - p_1 - d_2)$; вершине уровня 1 – значение

$$mS - (m-1)p_1 - (m-2)p_2 - \ldots - p_{m-1} - d_1 - d_2 - \ldots - d_m.$$

Данное значение является наименьшим значением суммарного запаздывания среди всех расписаний, при которых требование m является первым по порядку запаздывающим требованием.

Перейдём ко второй вершине первого уровня. В данной вершине на первую позицию упорядочивается требование $m+1$ (т.е. $j_1 = m+1$). Согласно лемме 5.6 эта вершина имеет два потомка: $j_2 = m-1$ и $j_2 = m$. При этом, дальнейшее ветвление в вершине, в которой на второе место ставится требование $m-1$, осуществлять не нужно, поскольку на предыдущих шагах (при обходе самой левой ветви дерева) необходимое ветвление уже было реализовано. Таким образом, мы производим «склейку» двух вершин дерева: вершины второго уровня самой левой ветви и рассматриваемой вершины. При ветвлении вершины $j_2 = m$ также получаем две новые вершины: $j_3 = m-2$ и $j_3 = m-1$, первая из которых ($j_3 = m-2$) склеивается с вершиной третьего уровня самой левой ветви и так далее. Получаем, что общее количество уникальных вершин (т.е. вершин, в которых осуществляется ветвление) в поддереве с корнем во второй вершине первого уровня равно m (все остальные вершины склеиваются с соответствующими вершинами самой левой ветви дерева).

Проводя рассуждения для вершин поддерева с корнем в k-ой вершине первого уровня, $1 \leq k \leq n - m + 1$, получаем, что данные вершины имеют k потомков, но только одна из дочерних вершин подлежит дальнейшему ветвлению, а остальные склеиваются с ранее построенными вершинами. Получаем, что в поддереве с корнем в любой вершине первого уровня имеется только m уникальных вершин. На рис. 5.2 приведено схематичное изображение дерева ветвлений.

Приведём формальное описание рассматриваемого алгоритма. Введём в рассмотрение три матрицы $\mathcal{J} = \{j_{\alpha,\beta}\}$, $\mathcal{F} = \{f_{\alpha,\beta}\}$ и $\mathcal{P} = \{p_{\alpha,\beta}\}$, где $\alpha \in \{1, 2, \ldots, m\}$ и $\beta \in \{m, m+1, \ldots, n\}$. Данные матрицы описывают все необходимые для построения оптимального расписания величины: $j_{\alpha,\beta}$ – требование, которое планируется к обслуживанию на позиции j_α при обходе поддерева с корнем в вершине β первого уровня дерева; $f_{\alpha,\beta}$ – наименьшее значение суммарного запаздывания требований, обслуживаемых на позициях $j_\alpha, j_{\alpha+1}, \ldots, j_m$, среди всех расписаний, при которых на позиции j_α обслужи-

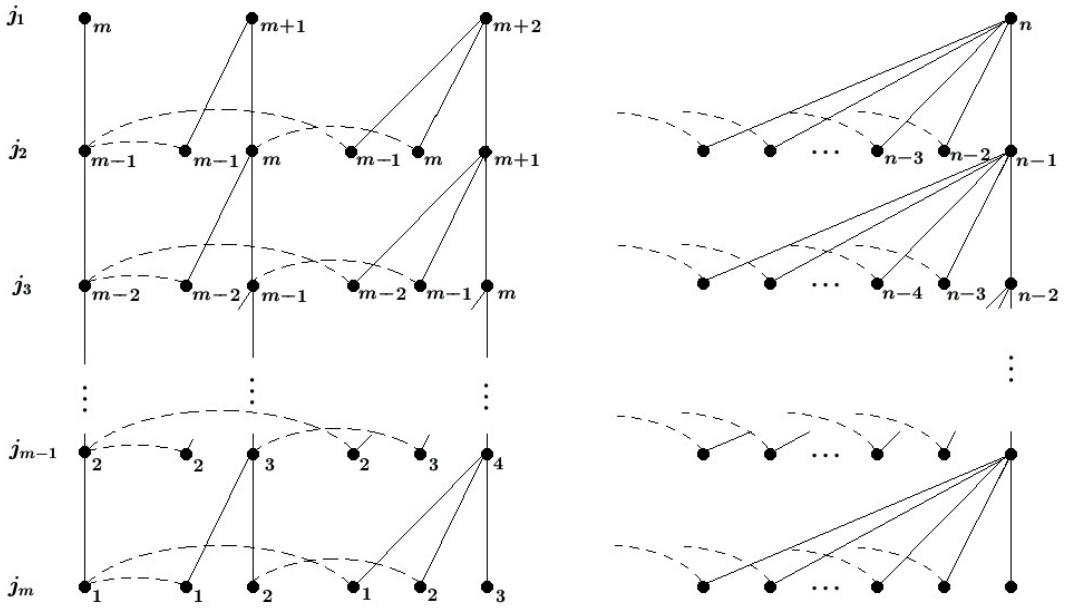

Рис. 5.2: Дерево ветвлений для предлагаемого алгоритма. Штрихованными линиями показано «склеивание» вершин. Пометки вершин означают требования, которые планируются на обслуживание в данных вершинах.

вается требование $j_{\alpha,\beta}$; $p_{\alpha,\beta}$ – суммарная продолжительность обслуживания требований, обслуживаемых на позициях $j_{\alpha+1}, \ldots, j_m$ при некотором оптимальном расписании. Пусть $S = t_0 + \sum_{j=1}^{n} p_j$.

Алгоритм 5.5 *FA*

1: $j_{m,m} := 1$, $f_{m,m} := S - d_1$, $p_{m,m} := 0$
2: **for** $\alpha = m-1, m-2, \ldots, 1$ **do**
3: $j_{\alpha,m} := m - \alpha + 1$; $f_{\alpha,m} := f_{\alpha+1,m} + S - p_{\alpha+1,m} - d_{m-\alpha+1}$;
$p_{\alpha,m} := p_{\alpha+1,m} + p_{m-\alpha+1}$
4: **end for**
5: **for** $\beta = m+1, m+2, \ldots, n$ **do**
6: $j_{m,\beta} := \beta - m + 1$; $f_{m,m} := S - d_{\beta-m+1}$; $p_{m,m} := 0$;
7: **for** $\alpha = m-1, m-2, \ldots, 1$ **do**
8: $i^* := \arg\min_{i=m,\ldots,\beta} \{f_{\alpha+1,i} + S - p_{\alpha+1,i} - d_{j_{\alpha,\beta}}\}$;
9: $j_{\alpha,\beta} := j_{\alpha+1,\beta} + 1$, $f_{\alpha,\beta} := f_{\alpha+1,i^*} + S - p_{\alpha+1,i^*} - d_{j_{\alpha,\beta}}$;
$p_{\alpha,\beta} := p_{\alpha+1,i^*} + p_{j_{\alpha,\beta}}$
10: **end for**
11: **end for**

После заполнения элементов матриц $\mathcal{J}$, $\mathcal{F}$ и $\mathcal{P}$ выполняется построение оптимального расписания следующим образом. Полагаем $\pi^* := \emptyset$. Затем находим среди вершин первого уровня дерева вершину с наименьшим значением $f_{1,\beta}$. Пусть это будет вершина β^*. Ставим требование j_{1,β^*} на первое

место при расписании π^*. Затем переходим к дочерним вершинам вершины β^*. Также находим среди них вершину с наименьшим значением $f_{2,\alpha}$ и соответствующее требование ставим на второе место при расписании π^* и так далее. После заполнения расписания π^* подобным образом, оставшиеся требования в произвольном порядке ставятся в начало расписания π^*.

Теорема 5.3 *Алгоритм FA (алгоритм 5.5) находит оптимальное расписание для данного случая задачи* $1 \,||\, \sum T_j$ *за* $O(n^3)$ *операций.*

Доказательство. Дерево обхода рассматриваемого алгоритма представляет собой полное дерево ветвлений, получаемое при использовании необходимых условий оптимальности, доказанных в леммах 5.5 и 5.6. Поскольку при обходе данного дерева указанным выше способом отрезаний веток не происходит (а происходит только склеивание вершин, основанное на свойствах рассматриваемого дерева), то оптимальное расписание будет в итоге получено. Каждая из $n-m+1$ вершин первого уровня имеет ровно m уникальных вершин. Обработка каждой уникальной вершины требует не более $O(n-m+1)$ операций. Следовательно, алгоритм FA находит оптимальное расписание в рассматриваемом случае за $O(n^2)$ операций. Максимум выпуклой функции $f(x) = (n-x+1)x$ достигается при $x = \frac{n+1}{2}$, а $f = \left(\frac{n+1}{2}\right) = \left(\frac{n+1}{2}\right)^2$. Следовательно трудоемкость алгоритма FA составляет $O(n^3)$ операций. □

Для заинтересованного читателя можно сформулировать следующую задачу: в исследуемой проблеме $1||\sum T_j$ количество запаздывающих требований не больше n. Запуская на решение алгоритм FA при $m = 0, 1, \ldots, n$ получим расписания $\pi_0, \pi_1, \ldots, \pi_n$. Мы можем проделать это за $O(n^4)$ операций. Выбираем среди этих $n+1$ расписаний расписание с наименьшим суммарным запаздыванием π^*. При каких ограничениях на исходные параметры проблемы $1||\sum T_j$ расписание π^* будет оптимальным, а при каких нет?

5.4 Метрика для задачи $1|r_j|\sum T_j$

Имеется множество N, состоящее из n требований, которые необходимо обслужить на одном приборе. Прибор готов начать обслуживание в момент времени $t_0 = 0$ и не может обслуживать более одного требования одновременно. Прерывания при обслуживании запрещены. Для каждого требования $j \in N$ заданы: момент поступления r_j, продолжительность обслуживания p_j и директивный срок d_j. Расписание $\pi = \{j_1, j_2, \ldots, j_n\}$ определяет порядок в

котором обслуживаются требования. Естественно рассматривать ранние расписания, при которых:

$$C_{j_1}(\pi) = max\{t_0, r_{j_1}\} + p_{j_1},$$

$$C_{j_k}(\pi) = max\{r_{j_k}, C_{j_{k-1}}(\pi)\} + p_{j_k}, k = 2, 3, \ldots, n,$$

где $C_j(\pi)$ – момент окончания обслуживания требования j при расписании π. Необходимо построить оптимальное расписание π^* минимизирующее целевую функцию – суммарное запаздывание $\sum\limits_{j \in N} T_j(\pi)$, где $T_j(\pi) = \max\{0, C_j(\pi) - d_j\}$ – запаздывание требования j при расписании π. В дальнейшем, если из контекста ясно о каком расписании идет речь, зависимость от π может опускаться. Данная задача является NP—трудной [114] и обозначается $1|r_j|\sum T_j$ [125].

Задача $1|r_j|\sum T_j$ полностью характеризуется $3n$ параметрами – директивными сроками, продолжительностями обслуживания и моментами поступления каждого из n требований. Будем говорить, что задан пример A задачи, если задано $3n$ параметров $\{r_j^A, p_j^A, d_j^A, j = 1, 2, \ldots, n\}$, характеризующих задачу.

Для частного случая $r_j = 0, j \in N$, задачи минимизации суммарного запаздывания, ранее был предложен полиномиальный приближенный алгоритм сложности $O(\frac{n^7}{\epsilon})$ операций [150], также для этого случая известен псевдополиномиальный алгоритм сложности $O(n^4 \sum p_j)$ [150]. В случае, если

$$p_1 \geq p_2 \geq \ldots \geq p_n,$$

$$d_1 \leq d_2 \leq \ldots \leq d_n,$$

сложность псевдополиномиального алгоритма может быть уменьшена до $O(n^2 \sum p_j)$ [166]. Для случая $1|r_j, p_j = p|\sum T_j$ известен полиномиальный алгоритм сложности $O(n^7)$ операций, предложенный P. Baptiste [82].

Далее предлагается подход приближенного решения задачи $1|r_j|\sum T_j$ с гарантированной погрешностью, основанный на введении метрики для пространства параметров задачи, и рассматриваются возможности применения данного подхода к другим задачам теории расписаний. Затем описываются численные эксперименты, проведенные для проверки предложенного метода.

5.4.1 Метрика для пространства параметров

Задача $1|r_j|\sum T_j$ полностью характеризуется $3n$ параметрами, что позволяет нам рассматривать примеры задачи, как точки в $3n-$мерном пространстве параметров $\Omega = \{r_1, \ldots, r_n, p_1, \ldots, p_n, d_1, \ldots, d_n\}$.

Лемма 5.7 *Пусть примеры A и B имеют одинаковые продолжительности обслуживания и директивные сроки:*

$$p_j^A = p_j^B, d_j^A = d_j^B, j \in N.$$

Тогда для любого расписания π,

$$|\sum_{j\in N} T_j^A(\pi) - \sum_{j\in N} T_j^B(\pi)| \le n \max_{j\in N} |r_j^A - r_j^B|. \tag{5.7}$$

Доказательство. Используя определение запаздывания и известное неравенство

$$|\max\{a,b\} - \max\{c,d\}| \le \max\{|a-c|, |b-d|\}, \forall a,b,c,d \in \mathbb{R}, \tag{5.8}$$

получим

$$|\sum_{j\in N} T_j^A - \sum_{j\in N} T_j^B| \le \sum_{j\in N} |C_j^A - C_j^B + d_j^B - d_j^A| \le \sum_{j\in N} |C_j^A - C_j^B| + \sum_{j\in N} |d_j^A - d_j^B|. \tag{5.9}$$

Или, учитывая равенство директивных сроков,

$$|\sum_{j\in N} T_j^A - \sum_{j\in N} T_j^B| \le \sum_{j\in N} |C_j^A - C_j^B|. \tag{5.10}$$

Учитывая свойства раннего расписания, заметим, что

$$|C_{j_1}^A - C_{j_1}^B| = |r_{j_1}^A - r_{j_1}^B| \le \max_{j\in N} |r_j^A - r_j^B|,$$

$$|C_{j_k}^A - C_{j_k}^B| \le \max\{|r_{j_k}^A - r_{j_k}^B|,\ |C_{j_{k-1}}^A - C_{j_{k-1}}^B|\} \le \max_{j\in N} |r_j^A - r_j^B|,$$

$$k = 2, \ldots, n.$$

Отсюда и из неравенства (5.10), получаем утверждение леммы. □

Лемма 5.8 *Пусть примеры A и B имеют одинаковые моменты поступления и директивные сроки:*

$$r_j^A = r_j^B, d_j^A = d_j^B, j \in N.$$

Тогда для любого расписания π,

$$|\sum_{j\in N} T_j^A(\pi) - \sum_{j\in N} T_j^B(\pi)| \le n \sum_{j\in N} |p_j^A - p_j^B|. \tag{5.11}$$

Доказательство. При условиях леммы также выполняется неравенство (5.10)

$$|\sum_{j\in N} T_j^A - \sum_{j\in N} T_j^B| \le \sum_{j\in N} |C_j^A - C_j^B|.$$

Учитывая свойства раннего расписания и равенство моментов поступления, получим

$$|C_{j_1}^A - C_{j_1}^B| = |p_{j_1}^A - p_{j_1}^B| \le \sum_{j\in N} |p_j^A - p_j^B|,$$

$$|C_{j_k}^A - C_{j_k}^B| \le |p_{j_k}^A - p_{j_k}^B| + |C_{j_{k-1}}^A - C_{j_{k-1}}^B| \le \sum_{j\in N} |p_j^A - p_j^B|, k = 2, \dots, n.$$

Отсюда и из неравенства (5.10) получаем утверждение леммы. □

Лемма 5.9 *Пусть примеры A и B имеют одинаковые моменты поступления и продолжительности обслуживания:*

$$r_j^A = r_j^B,\ p_j^A = p_j^B,\ j \in N.$$

Тогда для любого расписания π,

$$|\sum_{j\in N} T_j^A(\pi) - \sum_{j\in N} T_j^B(\pi)| \le \sum_{j\in N} |d_j^A - d_j^B|. \tag{5.12}$$

Доказательство. При условиях леммы $C_{j_k}^A = C_{j_k}^B, k \in N$, и неравенство (5.9) имеет вид

$$|\sum_{j\in N} T_j^A - \sum_{j\in N} T_j^B| \le \sum_{j\in N} |d_j^A - d_j^B|,$$

т.е. утверждение леммы выполняется. □

Теорема 5.4 *Функция, определенная на пространстве примеров $\Omega \times \Omega$,*

$$\rho(A,B) = n \max_{j\in N} |r_j^A - R_j^B| + n \sum_{j\in N} |p_j^A - p_j^B| + \sum_{j\in N} |d_j^A - d_j^B| \tag{5.13}$$

удовлетворяет аксиомам метрики.

Доказательство. Очевидно, что функция $\rho(A,B)$ симметрична и неотрицательна, причем $\rho(A,B) = 0$ тогда и только тогда, когда $A = B$. Неравенство треугольника выполняется в силу свойств модуля суммы. □

Лемма 5.10 *Для любых примеров A и B, и любого расписания π справедливо неравенство*

$$|\sum_{j\in N} T_j^A - \sum_{j\in N} T_j^B| \le \rho(A,B). \tag{5.14}$$

Доказательство. Пусть пример C имеет такие же моменты поступления и продолжительности обслуживания как и пример A, а директивные сроки как у примера B. Пусть далее, пример D имеет моменты поступления как у примера A, а директивные сроки и продолжительности обслуживания, как у примера B, тогда, используя леммы 5.7-5.9 получим

$$|\sum_{j\in N} T_j^A - \sum_{j\in N} T_j^B| \le$$

$$\le |\sum_{j\in N} T_j^B - \sum_{j\in N} T_j^D| + |\sum_{j\in N} T_j^D - \sum_{j\in N} T_j^C| + |\sum_{j\in N} T_j^A - \sum_{j\in N} T_j^C| \le$$

$$\le n \max_{j\in N} |r_j^A - r_j^B| + n\sum_{j\in N} |p_j^A - p_j^B| + \sum_{j\in N} |d_j^A - d_j^B| = \rho(A,B).$$

□

5.4.2 Метод изменения параметров

Теорема 5.5 *Пусть π^A и π^B – оптимальные расписания для примеров A и B, тогда*

$$\sum_{j\in N} T_j^A(\pi^B) - \sum_{j\in N} T_j^A(\pi^A) \le 2\rho(A,B). \tag{5.15}$$

Доказательство. Используя лемму 5.10, получим

$$\sum_{j\in N} T_j^A(\pi^B) - \sum_{j\in N} T_j^A(\pi^A) =$$

$$= (\sum_{j\in N} T_j^A(\pi^B) - \sum_{j\in N} T_j^B(\pi^B)) + (\sum_{j\in N} T_j^B(\pi^B) - \sum_{j\in N} T_j^B(\pi^A)) +$$

$$+(\sum_{j\in N} T_j^B(\pi^A) - \sum_{j\in N} T_j^A(\pi^A)) \le 2\rho(A,B).$$

□

Доказанная теорема позволяет использовать новый метод решения задачи $1|r_j|\sum T_j$, названный методом изменения параметров. Метод состоит

в том, чтобы использовать оптимальное расписание некоторого, полиномиально или псевдополиномиально решаемого примера B в качестве расписания для примера A. Теорема 5.5 позволяет оценить сверху погрешность такого решения при помощи функции $\rho(A,B)$. Естественно конструировать пример B так, чтобы минимизировать функцию $\rho(A,B)$. Таким образом, задача $1|r_j|\sum T_j$ заменяется задачей на минимизацию функции метрики.

Рассмотрим случай, когда искомый пример B должен принадлежать некоторому полиномиально или псевдополиномиально разрешимому классу примеров, задаваемому системой неравенств

$$\mathcal{A}\cdot R^B+\mathcal{B}\cdot P^B+\mathcal{C}\cdot D^B\le H,$$

где $R^B=(r_1^B,\ldots,r_n^B)^T$; $P^B=(p_1^B,\ldots,p_n^B)^T$; $D^B=(d_1^B,\ldots,d_n^B)^T$; $p_j^B\ge 0$; $r_j^B\ge 0$; $j\in N$; ^{T} – символ транспонирования; $\mathcal{A},\mathcal{B},\mathcal{C}$ – матрицы $m\times n$, и H – столбец из m элементов.

В этом случае задача минимизации функции метрики может быть поставлена как задача линейного программирования.

$$\min n\cdot(y^r-x^r)+n\cdot\sum_{j\in N}(y_j^p-x_j^p)+\sum_{j\in N}(y_j^d-x_j^d), \tag{5.16}$$

при условиях

$$\begin{cases} x^r\le r_j^A-r_j^B\le y^r,\\ x_j^p\le p_j^A-p_j^B\le y_j^p,\\ x_j^d\le d_j^A-d_j^B\le y^d,\\ r_j^B\ge 0, p_j^B\ge 0, j\in N,\\ \mathcal{A}\cdot R^B+\mathcal{B}\cdot P^B+\mathcal{C}\cdot D^B\le H. \end{cases}$$

Неизвестными в данной задаче являются $7n+2$ переменных:

$$r_j^B,p_j^B,d_j^B,x_j^p,y_j^p,x_j^d,y_j^d,x^r,y^r,j\in N.$$

Тем не менее, сепарабельность функции $\rho(A,B)$ во многих случаях позволяет находить её минимум гораздо проще, без использования методов линейного программирования.

5.4.3 Применение метода изменения параметров для других задач теории расписаний

Описанный метод не является жестко привязанным к виду целевой функции, что позволяет использовать его для решения других задач теории расписа-

ний. Теорему 5.5 можно обобщить на случай общего вида целевой функции $F(\pi)$.

Теорема 5.6 *Пусть $F(\pi)$ – некоторая регулярная целевая функция, а $\rho(A,B)$ – функция метрики, для любых A, B, π, удовлетворяющая неравенству*

$$|F^A(\pi) - F^B(\pi)| \leq \rho(A,B). \tag{5.17}$$

Пусть далее π^A и π^B – оптимальные расписания для примеров A и B, тогда

$$F^A(\pi^B) - F^A(\pi^A) \leq 2\rho(A,B). \tag{5.18}$$

Доказательство. Доказательство теоремы (5.6) повторяет доказательство теоремы (5.5) с заменой $\sum\limits_{j\in N} T_j$ на F. □

Таким образом для применения метода изменения параметров достаточно построить функцию $\rho(A,B)$, удовлетворяющую неравенству (5.17). Такие функции были построены ранее для задач $1||\sum T_j$ [10] и $1|r_j|L_{max}$ [49]. Здесь мы приведем вариант построения таких функций для общих случаев аддитивной и "минимаксной" целевой функции.

Лемма 5.11 *В случае аддитивной целевой функции вида*

$$F(\pi) = \sum_{j\in N} \phi_j(\pi, r_1, \ldots, r_n, p_1, \ldots, p_n, d_j), \tag{5.19}$$

функция

$$\rho(A,B) = \sum_{j\in N}\sum_{i\in N}(R_{ji}|r_j^A - r_j^B| + P_{ji}|p_j^A - p_j^B|) + \sum_{j\in N} D_j|d_j^A - d_j^B|, \tag{5.20}$$

удовлетворяет неравенству (5.17). Здесь R_{ji}, P_{ji} – константы Липшица для функции ϕ_i по переменным r_j и p_j, D_j – константа Липшица для функции ϕ_j по переменной $d_j, i, j \in N$. □

Лемма 5.12 *В случае "минимаксной" целевой функции вида*

$$F(\pi) = \max_{j\in N} \phi_j(\pi, r_1, \ldots, r_n, p_1, \ldots, p_n, d_j), \tag{5.21}$$

функция

$$\rho(A,B) = \sum_{j\in N}(R_j|r_j^A - r_j^B| + P_j|p_j^A - p_j^B|) + D\max_{j\in N}|d_j^A - d_j^B|, \tag{5.22}$$

удовлетворяет неравенству (5.17). Здесь R_j, P_j – наибольшие константы Липшица по переменным r_j и p_j из констант для функций ϕ_i, D – наибольшая из констант Липшица для функций ϕ_j по переменным $d_j, i, j \in N$. $\square$

Заметим, что функции (5.20) и (5.22) сепарабельны, что значительно облегчает нахождение их минимумов.

5.4.4 Численные эксперименты

Для определения эффективности предложенной схемы была проведена серия численных экспериментов. Классы в которых проводился поиск полиномиально разрешимых примеров представлены в таблице 5.2.

В первых трех классах решением является расписание, упорядоченное по неубыванию свободного параметра. Алгоритмы решения задач последних двух классов представлены в [82] и [33] и имеют сложности $O(n^7)$ и $O(n^4 \sum p_j)$ операций, соответственно.

Таблица 5.2: Классы примеров, использованные в численных экспериментах

Класс примеров	**Метрика между примером B класса и произвольным примером A**
$\{\mathcal{PR} : p_j = p, r_j = r, j \in N\}$	$\rho(A,B) = n \cdot \sum\limits_{j=1}^{n} \lvert p_j^A - p\rvert + n \cdot \max\limits_{j \in N} \lvert r_j^A - r\rvert$
$\{\mathcal{PD} : p_j = p, d_j = d, j \in N\}$	$\rho(A,B) = n \cdot \sum\limits_{j \in N} \lvert p_j^A - p\rvert + \sum\limits_{j \in N} \lvert d_j^A - d\rvert$
$\{\mathcal{RD} : r_j = r, d_j = d, j \in N\}$	$\rho(A,B) = n \cdot \max\limits_{j \in N} \lvert r_j^A - r\rvert + \sum\limits_{j \in N} \lvert d_j^A - d\rvert$
$\{\mathcal{P} : p_j = p, j \in N\}$	$\rho(A,B) = n \cdot \sum\limits_{j \in N} \lvert p_j^A - p\rvert$
$\{\mathcal{R}0 : r_j = 0, j \in N\}$	$\rho(A,B) = n \cdot \max\limits_{j \in N} \lvert r_j^A - r\rvert$

Для нахождения в указанных классах полиномиально разрешимого примера B, ближайшего к заданному примеру, необходимо найти минимум функций:

$$f(r) = n \cdot \max_{j \in N} |r_j^A - r|; \tag{5.23}$$

$$g(p) = n \cdot \sum_{j=1}^{n} |p_j^A - p|; \tag{5.24}$$

$$h(d) = \sum_{j \in N} |d_j^A - d|. \tag{5.25}$$

Лемма 5.13 *1) Минимум функции (5.23) достигается в точке* $r = \frac{r_{max}^A + r_{min}^A}{2}$, *где* $r_{max}^A = \max\limits_{j \in N} r_j^A$, $r_{min}^A = \min\limits_{j \in N} r_j^A$.

2) Минимум функции (5.24) достигается в некоторой точке $p \in \{p_1^A, \ldots, p_n^A\}$.

3) Минимум функции (5.25) достигается в некоторой точке $d \in \{d_1^A, \ldots, d_n^A\}$).

Доказательство. Функция $f(r)$ представима в виде

$$n \cdot \max_{j \in N} |r_j^A - r| = n \max\{r - r_{min}^A, r_{max}^A - r\} = n(\frac{r_{max}^A - r_{min}^A}{2} + |r - \frac{r_{max}^A + r_{min}^A}{2}|)$$

и очевидно имеет минимум в точке $\frac{r_{max}^A + r_{min}^A}{2}$.

Пусть функция $g(p)$ имеет минимум в точке p_0, тогда либо $f'(p_0) = 0$, либо $p_0 \in \{p_1^A, \ldots, p_n^A\}$). Поскольку $g(p)$ — кусочно-линейная функция, обращение её производной в нуль означает, что функция является константой на некотором интервале $[p_k^A, p_{k+1}^A], k = 1, \ldots, n-1$, а значит граничные точки p_k^A и p_{k+1}^A также являются точками минимума.

Последнее утверждение леммы о минимуме функции $h(d)$ доказывается аналогично. □

Было проведено несколько серий экспериментов. Во всех сериях использовались примеры с параметрами, распределенными равномерно на интервалах $[1, 100]$ для p_j^A, $[p_j, \sum\limits_{j \in N} p_j]$ для d_j^A и $[0, d_j - p_j]$ для r_j^A, $j \in N$. В первой серии экспериментов оценивалась величина различия между правой и левой частями неравенства (5.14). Данная величина позволяет оценить погрешность метода. Для каждого $n = 10, 20, \ldots, 100$ генерировалось 10000 пар примеров. Использованные в экспериментах расписания генерировались случайным образом. Для каждой пары вычислялась величина $\frac{|\sum\limits_{j \in N} T_j^A - \sum\limits_{j \in N} T_j^B|}{\rho(A,B}$. Также для определения параметров, имеющих наибольшее влияние на функцию метрики вычислялись процентные величины вкладов частей метрики, зависящих от продолжительностей обслуживания, директивных сроков и моментов поступления.

Результаты представлены в таблице 5.3. Среднее значение $\frac{|\sum\limits_{j \in N} T_j^A - \sum\limits_{j \in N} T_j^B|}{\rho(A,B}$ меняется от 5% до 10% при росте n, а части функции метрики, зависящие

от продолжительностей обслуживания, директивных сроков и моментов поступления дают вклады приблизительно $35\%, 20\%$ и 25% в общую величину функции, соответственно.

Таблица 5.3: Средняя разница между целевыми функциями и доли составных частей метрики

n	$\frac{\lvert\sum T_j^A - T_j^B\rvert}{\rho}$	$\frac{\rho_r}{\rho}$	$\frac{\rho_p}{\rho}$	$\frac{\rho_d}{\rho}$
10	11,7%	35,6%	42,3%	20,6%
20	10,4%	39,7%	39,4%	19,4%
40	8,9%	42,4%	37,4%	18,6%
60	7,8%	43,6%	36,6%	18,3%
80	7,3%	44,4%	34,4%	18%
100	6,7%	44,9%	35,7%	17,9%

Вторая серия экспериментов служила непосредственно проверке метода изменения параметров . Эксперименты проводились по следующей схеме. Рассматривались значения $n = 4, 5, \ldots, 10$, для каждого n генерировались по 10000 примеров. К каждому примеру применялась вышеописанная схема для нахождения приближенного решения со значением целевой функции F_e, затем при помощи алгоритма ветвей и границ искалось точное решение с оптимальным значением целевой функции F^*. Далее вычислялось Δ — отношение реальной погрешности схемы $\delta = F_e - F^*$ к её верхней оценке, определяемой неравенством (5.15):

$$\Delta = \frac{F_e - F^*}{2\rho(A, B)}. \tag{5.26}$$

Было обнаружено, что если поиск полиномиально разрешимого примера ведется в классе RD, то средняя погрешность решения растет от 20% до 30% от верхней оценки (5.15) при увеличении n, это показывает, что расписание по возрастанию продолжительностей обслуживание плохо применимо для примеров с выбранным распределением параметров. Для остальных классов средняя погрешность решения не зависит от n, и составляет несколько процентов от максимальной теоретической. Столь малая погрешность обусловлена тем, что примерно в 20% случаев метод изменения параметров давал точное решение задачи. Зависимость средней ошибки Δ от n представлена в таблице 5.4 [165].

Таблица 5.4: Средняя экспериментальная погрешность в процентах от теоретической

n	$\mathcal{PR}$	$\mathcal{PD}$	$\mathcal{RD}$	$\mathcal{P}$	$\mathcal{R}0$
4	2,5%	4,6%	20,8%	1,8%	2,9%
5	2,6%	4,8%	23,1%	1,9%	2,8%
6	2,6%	4,6%	24,6%	1,9%	2,7%
7	2,6%	4,7%	26%	1,9%	2,5%
8	2,5%	4,6%	27%	2%	2,3%
9	2,4%	4,7%	27,9%	2%	2,2%
10	2,4%	4,6%	28,6%	1,9%	2,1%

5.5 Канонические примеры задачи $1||\sum T_j$

В этом разделе будут описаны два NP-трудных случая задачи $1||\sum T_j$ – канонические примеры DL [114] и LG. NP-трудность канонических примеров задачи $1||\sum T_j$ доказана сведением NP-полной проблемы ЧЁТНО-НЕЧЁТНОЕ РАЗБИЕНИЕ к задаче $1||\sum T_j$.

5.5.1 Задача ЧЁТНО-НЕЧЁТНОЕ РАЗБИЕНИЕ (ЧНР).

Задано упорядоченное множество из $2n$ положительных целых чисел $B = \{b_1, b_2, \ldots, b_{2n}\}$, $b_i > b_{i+1}$, для всех $1 \leq i \leq 2n-1$. Требуется определить, существует ли разбиение множества B на два подмножества B_1 и B_2, такое, что выполняется $\sum\limits_{b_i \in B_1} b_i = \sum\limits_{b_i \in B_2} b_i$ и для каждой пары чисел $i = 1, 2, \ldots, n$ подмножество B_1 (следовательно, и B_2) содержит в точности один элемент из пары $\{b_{2i-1}, b_{2i}\}$.

Обозначим $\delta_i = b_{2i-1} - b_{2i},\ i = 1, \ldots, n$, $\delta = \sum\limits_{i=1}^{n} \delta_i$.

Построим модифицированный пример задачи **ЧЁТНО-НЕЧЁТНОЕ РАЗБИЕНИЕ:**

$$\begin{cases} a_{2n} = M + b; \\ a_{2i} = a_{2i+2} + b,\ i = n-1, \ldots, 1; \\ a_{2i-1} = a_{2i} + \delta_i,\ i = n, \ldots, 1, \end{cases} \tag{5.27}$$

где $b \gg n\delta,\ M \geq n^3 b$.

Очевидно выполняется $a_i > a_{i+1},\ i = 1, 2, \ldots, 2n-1$, кроме того, $\delta_i = b_{2i-1} - b_{2i} = a_{2i-1} - a_{2i},\ i = 1, \ldots, n$.

Лемма 5.14 *Исходный пример* **ЧНР** *имеет решение "ДА" тогда и только тогда, когда модифицированный пример* **ЧНР** *имеет решение "ДА".*

Доказательство. Пусть для исходного примера существует два подмножества B_1 и B_2, таких что $\sum\limits_{b_i \in B_1} b_i = \sum\limits_{b_i \in B_2} b_i$. Возьмем $A_1 = \{a_i | b_i \in B_1\}$, $A_2 = \{a_i | b_i \in B_2\}$. Тогда выполняется $\sum\limits_{a_i \in A_1} a_i = \sum\limits_{a_i \in A_2} a_i$.

Пусть для модифицированного примера существует два подмножества A_1 и A_2, таких что $\sum\limits_{a_i \in A_1} a_i = \sum\limits_{a_i \in A_2} a_i$. Возьмем $B_1 = \{b_i | a_i \in A_1\}$, $B_2 = \{b_i | a_i \in A_2\}$. Очевидно, $\sum\limits_{b_i \in B_1} b_i = \sum\limits_{b_i \in B_2} b_i$. □

5.5.2 Канонические DL-примеры задачи $1||\sum T_j$

Приведем полиномиальную схему сведения **ЧНР** к задаче $\mathbf{1 \mid\mid \sum T_j}$ [114]. Определим $\delta = \sum\limits_{i=1}^{n} (b_{2i-1} - b_{2i})$.

Пусть $a_{2i-1} = b_{2i-1} + (9n^2 + 3n - i + 1)\frac{1}{2}\delta + 5n(b_1 - b_{2n})$ и $a_{2i} = b_{2i} + (9n^2 + 3n - i + 1)\frac{1}{2}\delta + 5n(b_1 - b_{2n})$, $i = 1, \ldots, n$.

Построим *канонический пример* задачи $\mathbf{1 \mid\mid \sum T_j}$ для множества из $3n+1$ требований $N = \{V_1, V_2 \ldots, V_{2n}, W_1, W_2, \ldots, W_{n+1}\}$. Пусть $b = (4n+1)\frac{1}{2}\delta$. Зададим параметры требований следующим образом:

$$p_{V_i} = a_i, \qquad i = 1, 2, \ldots, 2n,$$

$$p_{W_i} = b, \qquad i = 1, 2, \ldots, n+1,$$

$$d_{V_i} = \begin{cases} (j-1)b + \frac{1}{2}\delta + (a_2 + a_4 + \ldots + a_{2i}), & \text{если } i = 2j-1, \\ d_{V_{2j-1}} + 2(n-j+1)(a_{2j-1} - a_{2j}), & \text{если } i = 2j, \\ & \text{где } j = 1, 2, \ldots, n, \end{cases}$$

$$d_{W_i} = \begin{cases} ib + (a_2 + a_4 + \ldots + a_{2i}), & \text{если } i = 1, 2, \ldots, n, \\ d_{W_n} + \frac{1}{2}\delta + b, & \text{если } i = n+1. \end{cases}$$

Пусть $\{V_{i,1}, V_{i,2}\} = \{V_{2i-1}, V_{2i}\}$, $i = 1, \ldots, n$. Определим *каноническое расписание* как расписание вида

$$\pi = (V_{1,1}, W_1, V_{2,1}, W_2, .., W_{n-1}, V_{n,1}, W_n, W_{n+1}, V_{n,2}, V_{n-1,2}, .., V_{1,2}).$$

Теорема 5.7 [114] *Для примеров задачи* $\mathbf{1 \mid\mid \sum T_j}$, *к которым сводится* **ЧНР**, *существует оптимальное расписание, которое является каноническим.* □

Если $\frac{1}{2}\delta \notin Z$, рассмотрим исходный пример **ЧНР**, где все b_i умножены на 2. Несложно показать эквивалентность исходного и нового примера.

Введём следующие обозначения: $d_j(t) = d_j - d_{W_{n+1}} + t$, $j \in N$, $t \in R$. Пусть $\pi_l(t)$ и $F_l(t)$ оптимальное расписание и соответствующее ему значение суммарного запаздывания для параметрического примера для множества, состоящего из $3(n-l)+1$ требований $N_l = \{V_{2l-1}, V_{2l}, W_l, \ldots, V_{2n-1}, V_{2n}, W_n, W_{n+1}\}$ с директивными сроками окончания обслуживания $d_j(t)$, где $j = V_{2l-1}, V_{2l}, W_l, \ldots, V_{2n-1}, V_{2n}, W_n, W_{n+1}$, $l = n, \ldots, 1$.

Оптимальное расписание для исходного примера будет $\pi_1(d_{W_{n+1}})$. Заметим, что шаги основного цикла алгоритма выполняются только для каждого целого t из интервала, длина которого не превышает δ.

Алгоритм 5.6 B-1 канонический

1: **Вспомогательный шаг.** Положим $\pi_{n+1}(t) := (W_{n+1})$, $F_{n+1}(t) := \max\{0, b-t\}$ для $t \in T_{n+1} := [d_{W_{n+1}} - \sum_{i=1}^{n} a_{2i} - nb - \delta, d_{W_{n+1}} - \sum_{i=1}^{n} a_{2i} - nb]$;

2: **Основной шаг.**

3: **for all** $l = n, n-1, \ldots, 1$ и $t \in T_l := [d_{W_{n+1}} - \sum_{i=1}^{l-1} a_{2i} - (l-1)b - (\delta - \sum_{i=l-1}^{n} \delta_i), d_{W_{n+1}} - \sum_{i=1}^{l-1} a_{2i} - (l-1)b]$ **do**

4: $\pi^1 := (V_{2l-1}, W_l, \pi_{l+1}(t - a_{2l-1} - b), V_{2l})$;
$\pi^2 := (V_{2l}, W_l, \pi_{l+1}(t - a_{2l} - b), V_{2l-1})$;
$F(\pi^1) := \max\{0, a_{2l-1} - d_{V_{2l-1}}(t)\} + \max\{0, a_{2l-1} + b - d_{W_l}(t)\} + F_{l+1}(t - a_{2l-1} - b) + \max\{0, \sum_{j=l}^{n}(a_{2j-1} + a_{2j} + b) + b - d_{V_{2l}}(t)\}$;
$F(\pi^2) := \max\{0, a_{2l} - d_{V_{2l}}(t)\} + \max\{0, a_{2l} + b - d_{W_l}(t)\} + F_{l+1}(t - a_{2l} - b) + \max\{0, \sum_{j=l}^{n}(a_{2j-1} + a_{2j} + b) + b - d_{V_{2l-1}}(t)\}$;
$F_l(t) := \min\{F(\pi^1), F(\pi^2)\}$; $\pi_l(t) := \arg\min\{F(\pi^1), F(\pi^2)\}$;

5: **end for**

6: Алгоритм возвращает расписание $\pi_1(d_{W_{n+1}})$ и соответствующее значение суммарного запаздывания $F_1(d_{W_{n+1}})$.

Теорема 5.8 Алгоритм B-1 канонический *строит оптимальное каноническое расписание за время* $O(n\delta)$ *для канонических примеров задачи* $\mathbf{1} \,||\, \sum \boldsymbol{T_j}$, *к которым сводятся примеры* **ЧНР**. □

Проверку истинности данного утверждения оставим заинтересованному читателю...

Необходимо отметить, что всё семейство алгоритмов B-1 находит множество оптимальных расписаний для задач $\langle\{p_j, d_j(t)\}, 0\rangle$, когда t "пробегает" по временно́й оси, находится множество оптимальных расписаний для разных значений директивных сроков. Поэтому все алгоритмы типа B-1 будут псевдополиноминальными.

5.5.3 Канонические LG-примеры задачи $1||\sum T_j$

Показано, что частный случай **B-1** (5.28) [6] задачи минимизации суммарного запаздывания для одного прибора $1||\sum T_j$ является NP-трудным в обычном смысле.

Приведём полиномиальную схему сведения модифицированного примера **ЧНР** к частному случаю (5.28) задачи $\mathbf{1\,||\sum T_j}$.

Количество требований $2n+1$. Требования обозначим следующим образом:

$$V_1, V_2, V_3, V_4, \ldots, V_{2i-1}, V_{2i}, \ldots, V_{2n-1}, V_{2n}, V_{2n+1},$$

$N = \{1, 2, \ldots, 2n, 2n+1\}$. Для упрощения будем использовать следующие обозначения исходных параметров требований $p_{V_i} = p_i$, $d_{V_i} = d_i$, $T_{V_i} = T_i$, $C_{V_i} = C_i$, $i = 1, \ldots, 2n+1$. Пример, удовлетворяющий следующим ограничениям, будем называть *каноническим LG-примером*:

$$\begin{cases} p_1 > p_2 > \ldots > p_{2n+1}; & (5.28.1) \\ d_1 < d_2 < \ldots < d_{2n+1}; & (5.28.2) \\ d_{2n+1} - d_1 < p_{2n+1}; & (5.28.3) \\ p_{2n+1} = M = n^3 b; & (5.28.4) \\ p_{2n} = p_{2n+1} + b = a_{2n}; & (5.28.5) \\ p_{2i} = p_{2i+2} + b = a_{2i},\ i = n-1, \ldots, 1; & (5.28.6) \\ p_{2i-1} = p_{2i} + \delta_i = a_{2i-1},\ i = n, \ldots, 1; & (5.28.7) \\ d_{2n+1} = \sum\limits_{i:=1}^{n} p_{2i} + p_{2n+1} + \frac{1}{2}\delta; & (5.28.8) \\ d_{2n} = d_{2n+1} - \delta; & (5.28.9) \\ d_{2i} = d_{2i+2} - (n-i)b + \delta,\ i = n-1, \ldots, 1; & (5.28.10) \\ d_{2i-1} = d_{2i} - (n-i)\delta_i - \varepsilon\delta_i,\ i = n, \ldots, 1, & (5.28.11) \end{cases} \tag{5.28}$$

где $b = n^2\delta$, $0 < \varepsilon < \frac{\min\limits_i \delta_i}{\max\limits_i \delta_i}$.

Директивные сроки примера (5.28) представлены на рис. 5.3.

Обозначим через $L = \frac{1}{2}\sum\limits_{i:=1}^{2n} p_i$, тогда $d_{2n+1} = L + p_{2n+1}$, так как $\frac{1}{2}\sum\limits_{i:=1}^{2n} p_i = \sum\limits_{i:=1}^{n} p_{2i} + \frac{1}{2}\delta$.

Необходимо отметить, что канонические примеры DL из работы [114] не удовлетворяют случаю (5.28).

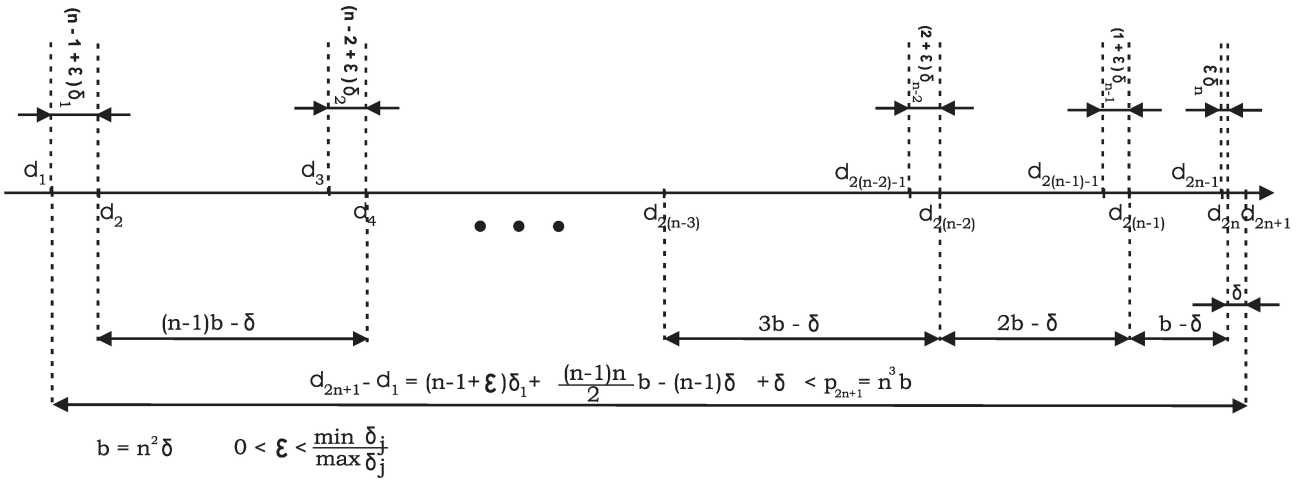

Рис. 5.3: Директивные сроки канонического LG-примера

5.5.4 Свойства канонических LG примеров задачи $1||\sum T_j$

Лемма 5.15 *Для случая (5.28) при любом расписании количество запаздывающих требований равно или n, или $(n+1)$.*

Доказательство.

1. Рассмотрим подмножество требований N' состоящее из $(n+2)$ самых коротких по продолжительности обслуживания требований и упорядочим их вначале расписания. Очевидно, что $\sum\limits_{i\in N'} p_i > (n+2)p_{\min} = (n+2)n^3b$, где $p_{\min} = \min\limits_{j\in N}\{p_j\} = p_{2n+1}$.

   Из (5.28.4)–(5.28.8) найдём

   $$d_{\max} = \max_{j\in N}\{d_j\} = d_{2n+1} = (n+1)n^3b + (b+2b+\ldots+nb) + \frac{1}{2}\delta,$$

   для которого будет выполняться

   $$d_{\max} = d_{2n+1} = (n+1)n^3b + \frac{n(n+1)}{2}b + \frac{1}{2}\delta < (n+2)n^3b < \sum_{i\in N'} p_i,$$

   т.е. требование обслуживаемое $(n+2)$ по порядку будет запаздывать при любом расписании. Все последующие требования будут также запаздывать, т.к. согласно (5.28.3) разность между директивными сроками любых двух требований меньше продолжительности обслуживания каждого требования. Таким образом, при любом расписании π количество незапаздывающих требований не превышает $(n+1)$.

2. Рассмотрим подмножество N'' состоящее из n самых длинных по продолжительности обслуживания требований и упорядочим их вначале расписания. Возможны два случая:

а) пусть $n = 2k$, тогда $N'' = \{V_1, V_2, \ldots, V_{2k-1}, V_{2k}\}$, будем иметь

$$P(N'') = nn^3b + 2(nb + (n-1)b + \ldots + (n-k+1)b) + \sum_{i:=1}^{k} \delta_i =$$

$$= nn^3b + 2\left(\frac{n(n+1)}{2} - \frac{(n-k)(n-k+1)}{2}\right)b + \sum_{i:=1}^{k} \delta_i.$$

Из (5.28.8)–(5.28.11) будем иметь

$d_{\min} = \min\limits_{j \in N}\{d_j\} = d_1 = d_{2n+1} - (\sum\limits_{i:=1}^{n-1}((n-i)b - \delta) + \delta + (n-1)\delta_1 - \varepsilon\delta_1) =$

$(n+1)n^3b + (b + 2b + \ldots + nb) + \frac{1}{2}\delta - (\sum\limits_{i:=1}^{n-1}((n-i)b - \delta) + \delta + (n-1)\delta_1 + \varepsilon\delta_1) >$

$P(N'')$;

б) пусть $n = 2k+1$, тогда $N'' = \{V_1, V_2, \ldots, V_{2k-1}, V_{2k}, V_{2(k+1)-1}\}$ и

$$P(N'') = nn^3b + 2(nb + (n-1)b + \ldots + (n-k+1)b) + (n-k)b + \sum_{i:=1}^{k+1} \delta_i =$$

$$= nn^3b + 2(\frac{n(n+1)}{2} - \frac{(n-k)(n-k+1)}{2})b + (n-k)b + \sum_{i:=1}^{k+1} \delta_i,$$

$d_{\min} = d_1 = d_{2n+1} - (\sum\limits_{i:=1}^{n-1}((n-i)b - \delta) + \delta + (n-1)\delta_1 - \varepsilon\delta_1) = (n+1)n^3b +$

$(b + 2b + \ldots + nb) + \frac{1}{2}\delta - (\sum\limits_{i:=1}^{n-1}((n-i)b - \delta) + \delta + (n-1)\delta_1 + \varepsilon\delta_1) > P(N'')$,

т.е. при любом расписании первые n требований будут не запаздывать. Таким образом, при любом расписании π количество незапаздывающих требований больше или равно n.

Следовательно, для случая (5.28) при любом расписании обслуживания требований множества N количество запаздывающих требований равно или n, или $(n+1)$. □

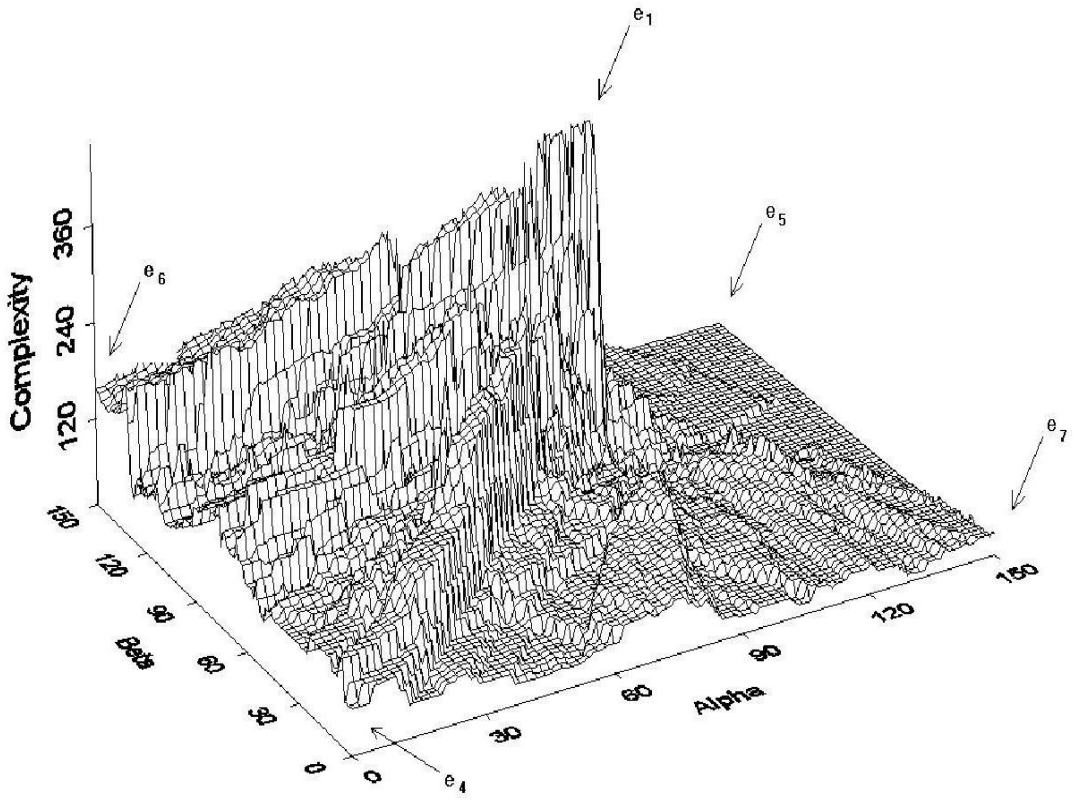

Рис. 5.4: Трудоемкость алгоритма SBA (вид справа)

Канонические примеры LG представлены на рисунках 5.4 и 5.5, где трудоемкость оценивается алгоритмами A и SBA.

Интересным на наш взгляд является вопрос: при каких ограничениях на параметры исходных данных алгоритм FA (алгоритм 5.5) при $m = n$ и $m = n + 1$) находит оптимальное решение канонических LG примеров

Лемма 5.16 *Если (5.28), то для каждого расписания вида* $\pi = (\pi_1, \pi_2)$ *существует расписание вида* $\pi' = (\pi_{EDD}, \pi_{SPT})$, *где* $\{\pi_1\} = \{\pi_{EDD}\}$, $\{\pi_2\} = \{\pi_{SPT}\}$, $|\{\pi_1\}| = n + 1$, $|\{\pi_2\}| = n$. *Причём,* $F(\pi) \geq F(\pi')$.

Доказательство. Рассмотрим частичное расписание π_1. Так как n первых требований при π_1 не запаздывают, а может запаздывать лишь последнее в этом расписании требование, то EDD порядок обслуживания требований множества $\{\pi_1\}$ будет оптимальным для этого подмножества. В этом случае на $(n + 1)$ месте будет обслуживаться требование $j = \arg\max\{d_i : i \in \{\pi_1\}\}$.

Рассмотрим частичное расписание π_2. Так как все n требований при π_2 запаздывают, то порядок SPT обслуживания требований множества $\{\pi_2\}$

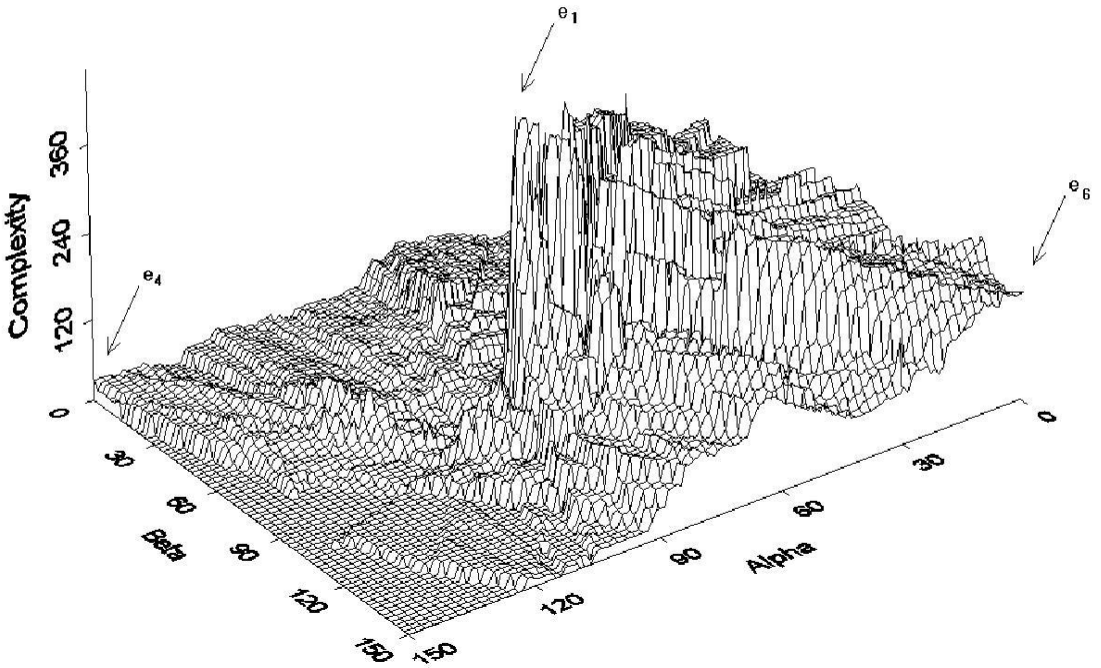

Рис. 5.5: Трудоемкость алгоритма SBA (вид слева)

будет оптимальным. □

Расписание вида

$$(V_{1,1}, V_{2,1}, \ldots, V_{i,1}, \ldots, V_{n,1}, V_{2n+1}, V_{n,2}, \ldots, V_{i,2}, \ldots, V_{2,2}, V_{1,2})$$

будем называть *каноническим LG-расписанием*, где $\{V_{i,1}, V_{i,2}\} = \{V_{2i-1}, V_{2i}\}$, $i = 1, 2, \ldots, n$.

Лемма 5.17 *Если расписание $\pi = (\pi_1, \pi_2)$, для которого $|\{\pi_1\}| = (n+1)$, $|\{\pi_2\}| = n$, неканоническое LG или не может быть преобразовано к каноническому LG расписанию применением правил EDD и SPT к частичным расписаниям π_1 и π_2, соответственно, то при расписании π некоторая пара требований $\{V_{2i-1},\ V_{2i}\}$, $i < n$, не запаздывает или расписание π имеет вид*

$$(V_{1,1}, \ldots, V_{i,1}, \ldots, V_{n-1,1}, V_{2n-1}, V_{2n}, V_{2n+1}, V_{n-1,2} \ldots, V_{i,2}, \ldots, V_{1,2}), \qquad (5.29)$$

т.е. пара требований $\{V_{2n-1}, V_{2n}\}$ обслуживается до требования V_{2n+1}, одно требование из каждой пары $\{V_{2i-1}, V_{2i}\}$, $i = 1, \ldots, n-1$, обслуживается до

требования V_{2n+1}, другое требование из этой пары после требования V_{2n+1}, а требования V_{2n+1} обслуживается $(n+2)$ по порядку.

Доказательство. Рассмотрим некоторое расписание вида $\pi = (\pi_1, \pi_2)$, где $|\{\pi_1\}| = (n+1),\ |\{\pi_2\}| = n$. Возможны следующие случаи.

1. Если $\{\pi_2\} = \{V_{1,2}, \ldots, V_{n,2}\}$, т.е. при π_2 упорядочены n требований по одному требованию из всех n пар $\{V_{2i-1}, V_{2i}\},\ i = 1, \ldots, n$. Упорядочим требования из π_2 по правилу SPT, а требования из π_1 по правилу EDD. Тогда получим каноническое расписание π', причём $F(\pi') \leq F(\pi)$, согласно лемме 5.16.

2. Если $\{\pi_2\} \neq \{V_{1,2}, \ldots, V_{n,2}\}$, то возможны ситуации.

   а) $V_{2n+1} \in \{\pi_2\}$,

   б) существует пара требований $\{V_{2j-1}, V_{2j}\} \subset \{\pi_2\}$.

   Тогда, учитывая что $|\{\pi_2\}| = n$, для некоторого i будем иметь $\{V_{2i-1}, V_{2i}\} \subset \{\pi_1\}$. □

Далее в теореме 5.9 будет показано, что для случая (5.28) все оптимальные расписания являются каноническими LG-расписаниями. Доказано, что произвольное неканоническое LG-расписание π может быть преобразовано к каноническому LG-расписанию π', причём $F(\pi) > F(\pi')$. При доказательстве теоремы используются выводы лемм 5.18–5.21.

Лемма 5.18 *Пусть расписание π имеет вид (5.29), при котором требование V_{2n+1} обслуживается на $(n+2)$ по порядку месте. Тогда для канонического LG-расписания*

$$\pi' = (V_{1,1}, \ldots, V_{i,1}, \ldots, V_{n-1,1}, V_{2n-1}, V_{2n+1}, V_{2n}, V_{n-1,2} \ldots, V_{i,2}, \ldots, V_{1,2})$$

выполняется $F(\pi) > F(\pi')$.

Доказательство. При расписании π требование V_{2n-1} обслуживается на позиции n "слева". Согласно лемме 5.17 требование V_{2n-1} не запаздывает, т.к. оно обслуживается n-м по порядку. Требование V_{2n+1} обслуживается на позиции $n+2$ "слева", поэтому запаздывает.

Очевидно, будем иметь для требований подмножества $\{V_2, V_4, \ldots, V_{2i}, \ldots, V_{2n-2}, V_{2n-1}\}$

$$P(\{V_2, V_4, \ldots, V_{2i}, \ldots, V_{2n-2}, V_{2n-1}\}) = nn^3b + \sum_{k:=1}^{n} kb + \delta_n =$$

$$= d_{2n+1} - n^3 b - \frac{1}{2}\delta + \delta_n,$$

согласно (5.28.8). Далее,

$$P(\{V_{1,1}, V_{2,1}, \ldots, V_{i,1}, \ldots, V_{n-1,1}, V_{2n-1}\}) + p_{2n} \geq$$
$$\geq P(\{V_2, V_4, \ldots, V_{2i}, \ldots, V_{2n-2}, V_{2n-1}\}) + p_{2n},$$

поэтому

$$C_{2n}(\pi) \geq d_{2n+1} + b - \frac{1}{2}\delta + \delta_n > d_{2n}.$$

Следовательно требование V_{2n}, которое обслуживается $(n+1)$ по порядку, при расписании π запаздывает.

Расписание π можно записать как $\pi = (\pi_{11}, V_{2n}, V_{2n+1}, \pi_{21})$. Рассмотрим следующее каноническое LG-расписание вида $\pi' = (\pi_{11}, V_{2n+1}, V_{2n}, \pi_{21})$. Покажем, что $F(\pi) > F(\pi')$.

а) Пусть при расписании π' требование V_{2n+1} не запаздывает. Тогда из (5.28.8) следует, что $d_{2n+1} - C_{2n+1}(\pi') \leq \frac{1}{2}\delta$, т.к. расписание π' каноническое.

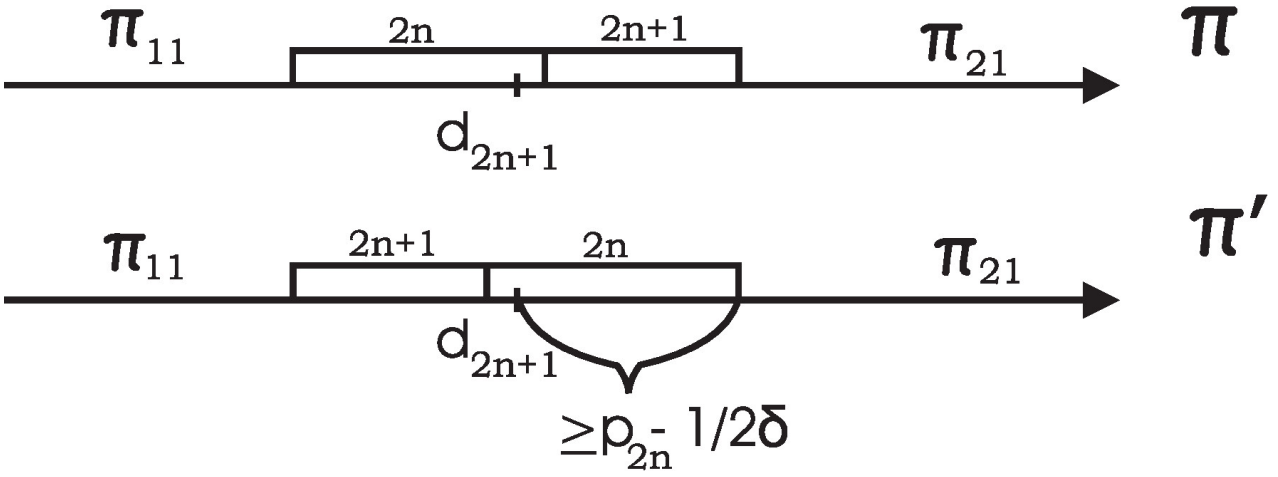

Рис. 5.6: Перестановка требований V_{2n} и V_{2n+1}

Из рис. 5.6 видно, что

$F(\pi) - F(\pi') = T_{2n}(\pi) + T_{2n+1}(\pi) - (T_{2n}(\pi') + T_{2n+1}(\pi')) = (T_{2n+1}(\pi) - T_{2n+1}(\pi')) - (T_{2n}(\pi') - T_{2n}(\pi)) \geq (p_{2n} - \frac{1}{2}\delta) - p_{2n+1} = p_{2n+1} + b - \frac{1}{2}\delta - p_{2n+1} > 0.$

б) Пусть при расписании π' требование V_{2n+1} запаздывает, тогда

$F(\pi) - F(\pi') = T_{2n}(\pi) + T_{2n+1}(\pi) - (T_{2n}(\pi') + T_{2n+1}(\pi')) = p_{2n} - p_{2n+1} = b > 0.$ □

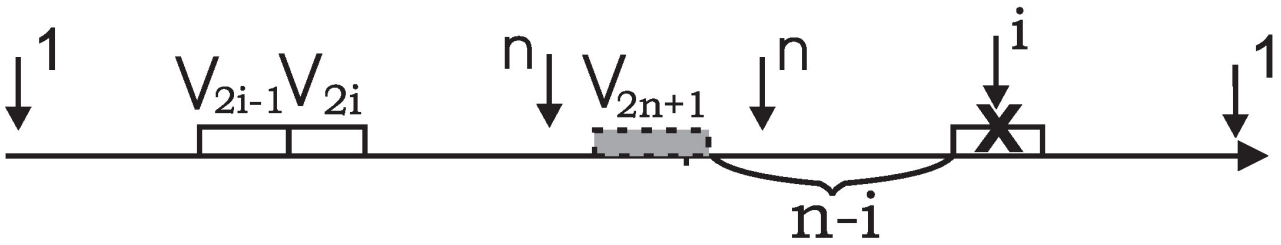

Рис. 5.7: Перестановка в неканоническом расписании

Лемма 5.19 *Пусть при расписании* $\pi = (\pi_{11}, V_{2i-1}, V_{2i}, \pi_{12}, \pi_{21}, X, \pi_{22})$ *пара требований* $\{V_{2i-1},\ V_{2i}\}$, $i < n$, *не запаздывает, а на позиции* i *"справа" обслуживается требование* $X \in \{V_{2j-1}, V_{2j}\}$, $j \geq i+1$. *Тогда для расписания* $\pi' = (\pi_{11}, V_{2i-1}, X, \pi_{12}, \pi_{21}, V_{2i}, \pi_{22})$ *выполняется* $F(\pi) > F(\pi')$.

Доказательство. Пусть при расписании π запаздывают требования только из множества $\{\pi_{21},\ X,\ \pi_{22}\}$, где $|\{\pi_{22}\}| = (i-1)$. Требование X занимает позицию i "справа" (см. рис. 5.7), в которой при каноническом расписании будет обслуживаться требование $V_{i,2} \in \{V_{2i-1}, V_{2i}\}$.

Построим расписание $\pi' = (\pi_{11}, V_{2i-1}, X, \pi_{12}, \pi_{21}, V_{2i}, \pi_{22})$. При обоих расписаниях запаздывает не меньше n требований. Поэтому при расписании π' перед требованием V_{2i} будет запаздывать не меньше, чем $n-i$ требований, и не больше, чем $n-i+1$ требований, согласно лемме 5.15. Поэтому

$$F(\pi) - F(\pi') \geq (p_{2i} - p_X)(n-i) - (d_X - d_{2i}).$$

а) Если $X = V_{2j}$, то $p_{2i} - p_X = (j-i)\ b$,

$$d_X - d_{2i} = \sum_{k:=i}^{j-1}(n-k)b - (j-i)\delta = n\ (j-i)\ b - \sum_{k:=i}^{j-1} kb - (j-i)\delta =$$

$$= n(j-i)b - i(j-i)b - \sum_{k:=0}^{j-1-i} kb - (j-i)\ \delta.$$

Следовательно,

$$F(\pi) - F(\pi') \geq (j-i)\ b\ (n-i) - (n\ (j-i)b - i(j-i)\ b - \sum_{k:=0}^{j-1-i} kb -$$

$$- (j-i)\ \delta) = \sum_{k:=0}^{j-1-i} kb + (j-i)\ \delta > 0.$$

б) Если $X = V_{2j-1}$, то $p_{2i} - p_X = ((j-i)\ b - \delta_j)$,

$$d_X - d_{2i} = \sum_{k:=i}^{j-1} (n-k)\ b - (j-i)\ \delta - (n-j)\ \delta_j - \varepsilon\delta_j =$$
$$= \ n(j-i)\ b - \sum_{k:=i}^{j-1} kb - (j-i)\ \delta - (n-j)\ \delta_j - \varepsilon\delta_j =$$
$$= n\ (j-i)\ b - i\ (j-i)\ b - \sum_{k:=0}^{j-1-i} kb - (j-i)\ \delta - (n-j)\ \delta_j - \varepsilon\delta_j.$$

Следовательно,

$$F(\pi) - F(\pi') \geq ((j-i)\ b - \delta_j)\ (n-i) - (n(j-i)\ b - -i\ (j-i)\ b - \sum_{k:=0}^{j-1-i} kb -$$
$$- (j-i)\ \delta - (n-j)\ \delta_j - \varepsilon\delta_j) = \sum_{k:=0}^{j-1-i} kb + (j-i)\ \delta - (j-i)\ \delta_j + \varepsilon\delta_j > 0.$$

□

Лемма 5.20 *Пусть при расписании* $\pi = (\pi_{11}, V_{2i-1}, V_{2i}, \pi_{12}, \pi_{21}, X, \pi_{22})$ *пара требований* $\{V_{2i-1},\ V_{2i}\}$, $i < n$, *не запаздывает, а на позиции* i *"справа" обслуживается требование* $X \in \{V_{2j-1}, V_{2j}\}$, $j < i-1$. *Тогда для расписания* $\pi' = (\pi_{11}, V_{2i-1}, X, \pi_{12}, \pi_{21}, V_{2i}, \pi_{22})$ *выполняется* $F(\pi) > F(\pi')$.

Доказательство. Пусть при расписании π запаздывают требования только из множества $\{\pi_{21},\ X,\ \pi_{22}\}$, где $|\{\pi_{22}\}| = (i-1)$. Требование X занимает позицию i "справа" (см. рис. 5.7), в которой при каноническом расписании будет обслуживаться требование $V_{i,2} \in \{V_{2i-1}, V_{2i}\}$.

Построим расписание $\pi' = (\pi_{11}, V_{2i-1}, X, \pi_{12}, \pi_{21}, V_{2i}, \pi_{22})$. При обоих расписаниях запаздывает не меньше n требований. Поэтому при расписании π' перед требованием V_{2i} будет запаздывать не меньше, чем $(n-i)$ требований (и не больше, чем $(n-i+1)$ требований), согласно лемме 5.15. Поэтому

$$F(\pi) - F(\pi') \geq (d_{2i} - d_X) - (p_X - p_{2i})(n-i+1).$$

а) Если $X = V_{2j}$, то $p_X - p_{2i} = (i-j)b$,

$$d_{2i} - d_{2j} = \sum_{k:=j}^{i-1} (n-k)b - (i-j)\delta = n(i-j)b - \sum_{k:=j}^{i-1} kb - (i-j)\delta =$$
$$= n(i-j)b - (i-1)(i-j)b + \sum_{k:=0}^{i-1-j} kb - (i-j)\delta.$$

Следовательно,

$$F(\pi) - F(\pi') \geq n\ (i-j)\ b - (i-1)\ (i-j)\ b + \sum_{k:=0}^{i-1-j} kb - (i-j)\ \delta -$$

$$- (i-j)\ b\ (n-i+1) = \sum_{k:=0}^{i-1-j} kb - (i-j)\delta > 0.$$

б) Если $X = V_{2j-1}$, то $p_X - p_{2i} = (i-j)\ b + \delta_j$,

$$d_{2i} - d_{2j-1} = \sum_{k:=j}^{i-1} (n-k)\ b - (i-j)\ \delta + (n-j)\ \delta_j + \varepsilon\delta_j =$$

$$= n\ (i-j)\ b - \sum_{k:=j}^{i-1} kb - (i-j)\ \delta + (n-j)\ \delta_j + \varepsilon\delta_j =$$

$$= n(i-j)\ b - (i-1)\ (i-j)\ b + \sum_{k:=0}^{i-1-j} kb - (i-j)\ \delta + (n-j)\ \delta_j + \varepsilon\delta_j.$$

Следовательно,

$$F(\pi) - F(\pi') \geq n(i-j)\ b - (i-1)\ (i-j)\ b + \sum_{k:=0}^{i-1-j} kb - (i-j)\ \delta +$$

$$+ (n-j)\ \delta_j + \varepsilon\delta_j - ((i-j)\ b + \delta_j)\ (n-i+1) =$$

$$= \sum_{k:=0}^{i-1-j} kb - (i-j)\ \delta - \delta_j + \varepsilon\delta_j > 0. \qquad \square$$

Лемма 5.21 *Пусть при расписании* $\pi = (\pi_{11}, V_{2i-1}, V_{2i}, \pi_{12}, \pi_{21}, X, \pi_{22})$ *пара требований* $\{V_{2i-1},\ V_{2i}\}$, $i < n$, *не запаздывает, а на позиции* i *“справа” обслуживается требование* $X \in \{V_{2(i-1)-1}, V_{2(i-1)}\}$. *Пусть при расписании* $\pi' = (\pi_{11}, V_{2i-1}, X, \pi_{12}, \pi_{21}, V_{2i}, \pi_{22})$ *требование* Y *занимает позицию* $(n+1)$ *и* $T_Y(\pi') < 2\delta$. *Тогда выполняется* $F(\pi) > F(\pi')$.

Доказательство. Пусть при расписании π запаздывают требования только из множества $\{\pi_{21},\ X,\ \pi_{22}\}$, где $|\{\pi_{22}\}| = (i-1)$. Требование X занимает позицию i “справа” (см. рис. 5.7), в которой при каноническом расписании будет обслуживаться требование $V_{i,2} \in \{V_{2i-1}, V_{2i}\}$.

Построим расписание $\pi' = (\pi_{11}, V_{2i-1}, X, \pi_{12}, \pi_{21}, V_{2i}, \pi_{22})$.

При расписании π перед требованием V_{2i} будет запаздывать не меньше $(n-i)$ требований. Поэтому

$$F(\pi) - F(\pi') > (d_{2i} - d_X) - (p_X - p_{2i})(n-i) - (T_Y(\pi') - T_Y(\pi)) >$$

$$> (d_{2i} - d_X) - (p_X - p_{2i})(n-i) - 2\delta.$$

Возможны ситуации:

a) Если $X = V_{2(i-1)}$, то $p_X - p_{2i} = b$, поэтому $d_{2i} - d_{2i-2} = (n-i+1)b - \delta$.

Следовательно

$F(\pi) - F(\pi') > (n-i+1)b - \delta - (n-i)b - 2\delta = b - 3\delta > 0.$

b) Если $X = V_{2(i-1)-1}$, то $p_X - p_{2i} = b + \delta_{i-1}$,

$d_{2i} - d_{2i-2} = (n-i+1)b - \delta + (n-i+1)\delta_{i-1} + \varepsilon\delta_{i-1}$.

Поэтому,

$F(\pi) - F(\pi') > (n-i+1)b - \delta + (n-i+1)\delta_{i-1} + \varepsilon\delta_{i-1} -$
$- (n-i)(b+\delta_{i-1}) - 2\delta = b - 3\delta + \delta_{i-1} + \varepsilon\delta_{i-1} > 0$, т.к. $b = n^2\delta$. □

Выводы леммы 5.21 используются при доказательстве теоремы 5.9. При этом имеет место $T_Y(\pi') < 2\delta$. Ситуация $T_Y(\pi') \geq 2\delta$ нами не рассматривается, т.к. она не встречается.

На основе полученных лемм докажем следующую теорему.

Теорема 5.9 *Для случая (5.28) все оптималъные расписания являются каноническими (или могут быть преобразованы к каноническим расписаниям применением правила EDD к $(n+1)$ требованиям, обслуживаемым вначале расписания).*

Доказательство. Пусть π – некоторое произвольное расписание. Согласно лемме 5.16 можно рассматривать (или преобразовать к нему) только расписание следующего вида $\pi = (\pi_{EDD}, \pi_{SPT})$, где $|\{\pi_{EDD}\}| = (n+1)$. При этом расписании требование V_{2n+1} занимает или позицию $(n+1)$, или позицию $(n+2)$. Пусть расписание π неканоническое.

Так как π не является каноническим расписанием, то в силу леммы 5.17 имеем при расписании π пару требований $\{V_{2i-1},\ V_{2i}\}$, которые не запаздывают, $i < n$, или расписание π имеет вид (5.29), при котором требование V_{2n+1} обслуживается на $(n+2)$ месте.

Если расписание имеет вид (5.29), то по лемме 5.18 для расписания $\pi' = (V_{1,1}, .., V_{i,1}, \ldots, V_{n-1,1}, V_{2n-1}, V_{2n+1}, V_{2n}, V_{n-1,2} \ldots, V_{i,2}, \ldots, V_{1,2})$, которое является каноническим, выполняется $F(\pi) > F(\pi')$. Переобозначим $\pi := \pi'$.

Далее опишем алгоритм, состоящий из двух циклов, преобразования исходного расписания π к каноническому виду.

Цикл 1. Пока среди незапаздывающих требований при очередном расписании π присутствует пара требований V_{2i-1}, V_{2i}, причём на позиции i "справа" обслуживается требование $X \notin \{V_{2(i-1)-1}, V_{2(i-1)}\}, X \neq V_{2n+1}$. Применяем для требований V_{2i} и X перестановку описанную в леммах 5.19 и 5.20. В результате значение целевой функции уменьшится.

Конец цикла 1.

Переобозначим $\pi := \pi'$. Количество шагов в цикле 1, очевидно, не превышает n. Первые $(n+1)$ требований при расписании π упорядочим по правилу EDD.

Требование V_{2n+1} занимает или позицию $(n+1)$ или позицию $(n+2)$ при расписании π. Если требование V_{2n+1} занимает позицию $(n+2)$ "слева", то позиции n и $(n+1)$ "слева" занимают требования V_{2n-1} и V_{2n}, соответственно, согласно **циклу 1** и EDD сортировке.

Рассмотрим возможные ситуации:

I. Пусть требование V_{2n+1} занимает позицию $(n+2)$. Рассматривается расписание следующего вида $\pi = (\pi_1, V_{2n-1}, V_{2n}, V_{2n+1}, \pi_2)$, при котором требование V_{2n} обслуживается $(n+1)$ по порядку. При частичных расписаниях π_1 и π_2 обслуживается по $(n-1)$ требованию, т.е. $|\{\pi_1\}| = n-1 = |\{\pi_2\}|$. Учитывая, что **циклом 1** не исключаются только ситуации описанные в лемме 5.21. Поэтому выполняется $P(\pi_1) + 2qb + \delta > P(\pi_2) > P(\pi_1) + 2qb - \delta$, где q – количество ситуаций описанных в лемме 5.21 при расписании π.

Примером рассматриваемой ситуации может служить расписание, при котором множества:
$\{\pi_1\} = \{V_{2i-1}, V_{2i}\} \cup \{V_{1,1}, V_{2,1}, \ldots, V_{i-2,1}, V_{i+1,1}, \ldots, V_{n-1,1}\}$;
$\{\pi_2\} = \{V_{2(i-1)-1}, V_{2(i-1)}\} \cup \cup \{V_{1,2}, V_{2,2}, \ldots, V_{i-2,2}, V_{i+1,2}, \ldots, V_{n-1,2}\}$.
Тогда $q = 1$ и $P(\pi_1) + 2b + \delta > P(\pi_2) > P(\pi_1) + 2b - \delta$, так как
$-(\delta - \delta_{i-1} - \delta_i - \delta_n) < P(\{V_{1,1}, V_{2,1}, \ldots, V_{i-2,1}, V_{i+1,1}, \ldots, V_{n-1,1}\}) -$
$- P(\{V_{1,2}, V_{2,2}, \ldots, V_{i-2,2}, V_{i+1,2}, \ldots, V_{n-1,2}\}) < \delta - \delta_{i-1} - \delta_i - \delta_n$
и выполняется $P(\{V_{2(i-1)-1}, V_{2(i-1)}\}) - P(\{V_{2i-1}, V_{2i}\}) = 2b + \delta_{i-1} - \delta_i$.

Рассмотрим две ситуации, когда $q = 1$ и $q > 1$. При $q = 0$ расписание π имеет вид (5.29) (см. лемму 5.18). Этот случай мы рассмотрели выше.

а) Пусть $q = 1$.

Известно, что $\sum\limits_{i:=1}^{2n+1} p_i = 2L + p_{2n+1} = 2L + n^3 b$.

Обозначим через $\Delta = P(\pi_2) - (P(\pi_1) + 2b)$,
для этой величины верно $-\delta < \Delta < \delta$.

Пусть $S = P(\pi_1)$. Тогда $2S + 2b + \Delta + p_{2n-1} + p_{2n} + p_{2n+1} = 2S + \Delta + 2b + 3n^3b + 2b + \delta_n = 2L + n^3b$.

Поэтому

$$L = S + \frac{1}{2}\Delta + 2b + n^3b + \frac{1}{2}\delta_n,$$

тогда

$$C_{2n}(\pi) = P(\pi_1) + p_{2n-1} + p_{2n} = S + 2n^3b + 2b + \delta_n =$$

$$= L + n^3b + \frac{1}{2}\delta_n - \frac{1}{2}\Delta.$$

Известно, что $L + n^3b = d_{2n+1}$,
тогда выполняется $-\delta < C_{2n}(\pi) - d_{2n+1} < \delta$.

Необходимо рассмотреть два подслучая, когда $C_{2n}(\pi) \geq d_{2n+1}$ и $C_{2n}(\pi) < d_{2n+1}$.

1. $C_{2n}(\pi) \geq d_{2n+1}$.
   Рассмотрим расписание $\pi' = (\pi_1, V_{2n-1}, V_{2n+1}, V_{2n}, \pi_2)$,
   для которого:
   $F(\pi) - F(\pi') = T_{2n}(\pi) + T_{2n+1}(\pi) - (T_{2n}(\pi') + T_{2n+1}(\pi')) =$
   $= (T_{2n+1}(\pi) - T_{2n+1}(\pi')) - (T_{2n}(\pi') =$
   $= (p_{2n+1} + (C_{2n}(\pi) - d_{2n+1})) - p_{2n+1} = C_{2n}(\pi) - d_{2n+1} \geq 0.$
2. $C_{2n}(\pi) < d_{2n+1}$.
   При этом, $C_{2n}(\pi) > d_{2n}$, т.к. $d_{2n+1} - d_{2n} = \delta$ и $d_{2n+1} - C_{2n}(\pi) < \delta$.

   Распишем структуру расписания π.

   $$\pi = (\pi_{11}, V_{2i-1}, V_{2i}, \pi_{12}, V_{2n-1}, V_{2n}, V_{2n+1}, \pi_{21}, X, \pi_{22}),$$

   где $|\{\pi_{22}\}| = i - 1$, $X \in \{V_{2(i-1)-1}, V_{2(i-1)}\}$.

   Если $X = V_{2(i-1)-1}$, то транспозиция соседних требований $V_{2(i-1)-1}$ и $V_{2(i-1)}$ согласно правилу SPT не увеличит значение целевой функции.
   Пусть $X = V_{2(i-1)}$. При расписании π запаздывает $(n+1)$ требование. Построим расписание

   $$\pi' = (\pi_{11}, V_{2i-1}, X, \pi_{12}, V_{2n-1}, V_{2n}, V_{2n+1}, \pi_{21}, V_{2i}, \pi_{22}).$$

   Выполняется

$F(\pi) - F(\pi') = (d_{2i} - d_{2(i-1)}) - (n-i+1)(p_{2(i-1)} - p_{2i}) =$
$= (n-i+1)b - \delta - (n-i+1)b = -\delta$, т.е. значение целевой функции увеличилось на δ. Тогда $C_{2n}(\pi') - d_{2n+1} > b - \delta$. Построим расписание

$$\pi'' = (\pi_{11}, V_{2i-1}, X, \pi_{12}, V_{2n-1}, V_{2n+1}, V_{2n}, \pi_{21}, V_{2i}, \pi_{22}).$$

Оценим разность значений целевой функции $F(\pi') - F(\pi'') > (p_{2n+1} + b - \delta) - p_{2n+1} > b - \delta$. Тогда $F(\pi) - F(\pi'') = b - \delta - \delta > 0$.

б) Пусть $q > 1$. Тогда $d_{2n} - C_{2n}(\pi) > b - 2\delta$.

Если $q = 2$, то при расписании π', рассмотренном в лемме 5.21, для требования $Y = V_{2n}$ выполняется $T_Y(\pi') < 2\delta$. Перестановка, описанная в лемме 5.21, уменьшит значение целевой функции.

Если $q > 2$, то при расписании π' будет запаздывать n требований, поэтому, согласно лемме 5.21, имеет место неравенство $F(\pi) > F(\pi')$.

II. Пусть требование V_{2n+1} обслуживается $(n+1)$ по порядку. То в ситуации, описанной в лемме 5.21, будем иметь $T_Y(\pi') = T_{2n+1}(\pi') < \frac{1}{2}\delta$. Преобразование расписания, согласно лемме 5.21, уменьшит значение целевой функции.

Цикл 2. Пока среди незапаздывающих требований при очередном расписании π присутствует пара требований V_{2i-1}, V_{2i}, причём на позиции i "справа" обслуживается требование $X \in \{V_{2(i-1)-1}, V_{2(i-1)}\}$. Применяем для требований X и V_{2i} перестановку, описанную в пунктах **I** и **II**. При этом значение целевой функции уменьшается.

Конец цикла 2. Конец алгоритма преобразования исходного неканонического расписания.

Таким образом, произвольное неканоническое расписание π за $O(n)$ операций можно преобразовать к каноническому расписанию π^*. Причём $F(\pi) > F(\pi^*)$. □

Теорема 5.10 *Решение задачи* **ЧНР** *будет "ДА" тогда и только тогда, когда при оптимальном каноническом расписании* $C_{2n+1}(\pi) = d_{2n+1}$.

Доказательство. Рассмотрим каноническое расписание вида

$$\pi = (V_{1,1}, V_{2,1}, \ldots, V_{i,1}, \ldots, V_{n,1}, V_{2n+1}, V_{n,2}, \ldots, V_{i,2}, \ldots, V_{2,2}, V_{1,2}).$$

Известно, что требования $V_{n,2}, \ldots, V_{i,2}, \ldots, V_{2,2}, V_{1,2}$ запаздывают. Требование V_{2n+1} может запаздывать, тогда $F(\pi) = \sum\limits_{i:=1}^{n} T_{V_{i,2}}(\pi) + T_{V_{2n+1}}(\pi)$.

Обозначим через $\sum\limits_{i:=1}^{2n+1} p_i = C$. Тогда

$$\sum_{i:=1}^{n} C_{V_{i,2}}(\pi) = nC - \sum_{i:=1}^{n-1}(n-i)p_{V_{i,2}}.$$

Обозначим

$$\phi(i) = \begin{cases} 1, \text{ если } V_{i,2} = V_{2i-1}, \\ 0, \text{ если } V_{i,2} = V_{2i}, \end{cases}$$

тогда

$$d_{V_{i,2}} = d_{2n+1} - \left(\sum_{k:=i}^{n-1}(n-k)b + (n-i+1)\delta + \phi(i)((n-i)\delta_i + \varepsilon\delta_i) \right),$$

поэтому

$$\sum_{i:=1}^{n} T_{V_{i,2}}(\pi) = nC - \sum_{i:=1}^{n-1}(n-i)p_{V_{i,2}} -$$

$$-\sum_{i:=1}^{n} \left(d_{2n+1} - \left(\sum_{k:=i}^{n-1}(n-k)b + (n-i+1)\delta + \phi(i)((n-i)\delta_i + \varepsilon\delta_i) \right) \right).$$

Задача $\min\limits_{\pi} F(\pi) = \min(\sum\limits_{i:=1}^{n} T_{V_{i,2}}(\pi) + T_{V_{2n+1}}(\pi))$ сводится к задаче максимизации функции Φ, где функция

$$\Phi = \sum_{i:=1}^{n-1}(n-i)p_{V_{i,2}} - \sum_{i:=1}^{n} \phi(i)((n-i)\delta_i + \varepsilon\delta_i) - T_{V_{2n+1}}(\pi) :$$

1. Если $V_{i,2} = V_{2i},\ i = 1, \ldots, n$, то $T_{V_{2n+1}}(\pi) = \frac{1}{2}\delta$, $\Phi_1 = \sum\limits_{i:=1}^{n-1}(n-i)p_{2i} - \frac{1}{2}\delta$.

2. Если $V_{i,2} = V_{2i-1},\ i = 1, \ldots, n$, то $T_{V_{2n+1}}(\pi) = \max\{-\frac{1}{2}\delta, 0\} = 0$,
$\Phi = \sum\limits_{i:=1}^{n-1}(n-i)p_{2i-1} - \sum\limits_{i:=1}^{n}((n-i)\delta_i + \varepsilon\delta_i) =$
$\sum\limits_{i:=1}^{n-1}(n-i)p_{2i} + \sum\limits_{i:=1}^{n-1}(n-i)\delta_i - \sum\limits_{i:=1}^{n}((n-i)\delta_i + \varepsilon\delta_i) = \Phi_1 + \frac{1}{2}\delta - \sum\limits_{i:=1}^{n}\varepsilon\delta_i.$

Функция Φ достигает максимальное значение равное $\Phi_1 + \frac{1}{2}\delta - \frac{1}{2}\sum\limits_{i:=1}^{n} \varepsilon\delta_i$, когда $\sum\limits_{i:=1}^{n} \phi(i)(\varepsilon\delta_i) = \frac{1}{2}\sum\limits_{i:=1}^{n} \varepsilon\delta_i$, что равнозначно $\sum\limits_{i:=1}^{n} \phi(i)\delta_i = \frac{1}{2}\sum\limits_{i:=1}^{n} \delta_i$. Следовательно для модифицированного примера существуют два подмножества A_1 и A_2, таких что $\sum\limits_{a_i \in A_1} a_i = \sum\limits_{a_i \in A_2} a_i$ (ответ "ДА"). При этом $C_{2n+1}(\pi) = d_{2n+1}$.

Если решение модифицированного примера **ЧНР** "НЕТ", то не выполняется равенство $\sum\limits_{i:=1}^{n} \phi(i)\delta_i = \frac{1}{2}\sum\limits_{i:=1}^{n} \delta_i$. Учитывая значение d_{2n+1}, будем иметь $C_{2n+1}(\pi) \neq d_{2n+1}$.

Если $C_{2n+1}(\pi) = d_{2n+1}$, то из этого следует, что выполняется $\sum\limits_{i:=1}^{n} p_{V_{i,1}} = \sum\limits_{i:=1}^{n} p_{V_{2i}} + \frac{1}{2}\delta = \sum\limits_{i:=1}^{n} p_{V_{i,2}}$, т.е. решение модифицированного примера **ЧНР** будет иметь ответ "ДА". □

5.6 Трудоёмкость алгоритмов

Будет показано, что трудоёмкость известных алгоритмов [24, 105, 208] решения задачи $1 \mid\mid \sum T_j$ не меньше $O(n2^{(n-1)/3-1})$ для *канонических DL-примеров* и не меньше $O(n2^{(n-1)/2})$ для примеров случая **BF**.

В основе известных алгоритмов [24, 105, 208, 209] используются правила сокращения перебора: **правила исключения 1–4** , **анализ параметров E_j, L_j, построение модифицированного примера**. Показано, что алгоритмы, использующие только эти правила сокращения перебора, имеют экспоненциальную трудоёмкость (от количества требований n) для *канонических* случаев задачи $1||\sum T_j$.

В подразделе исследуется трудоёмкость известных алгоритмов для *канонических примеров DL*. Предложен алгоритм трудоёмкости $O(n\delta)$ операций.

5.6.1 Канонические DL-примеры задачи $1||\sum T_j$

Далее будет показано, что алгоритмы поиска оптимального расписания, в которых используются известные методы сокращения перебора (правила исключения 1-4, использование E_j и L_j, построение модифицированного примера) [24, 208], в случае *канонических DL-примеров* имеют экспоненциальную трудоёмкость от n.

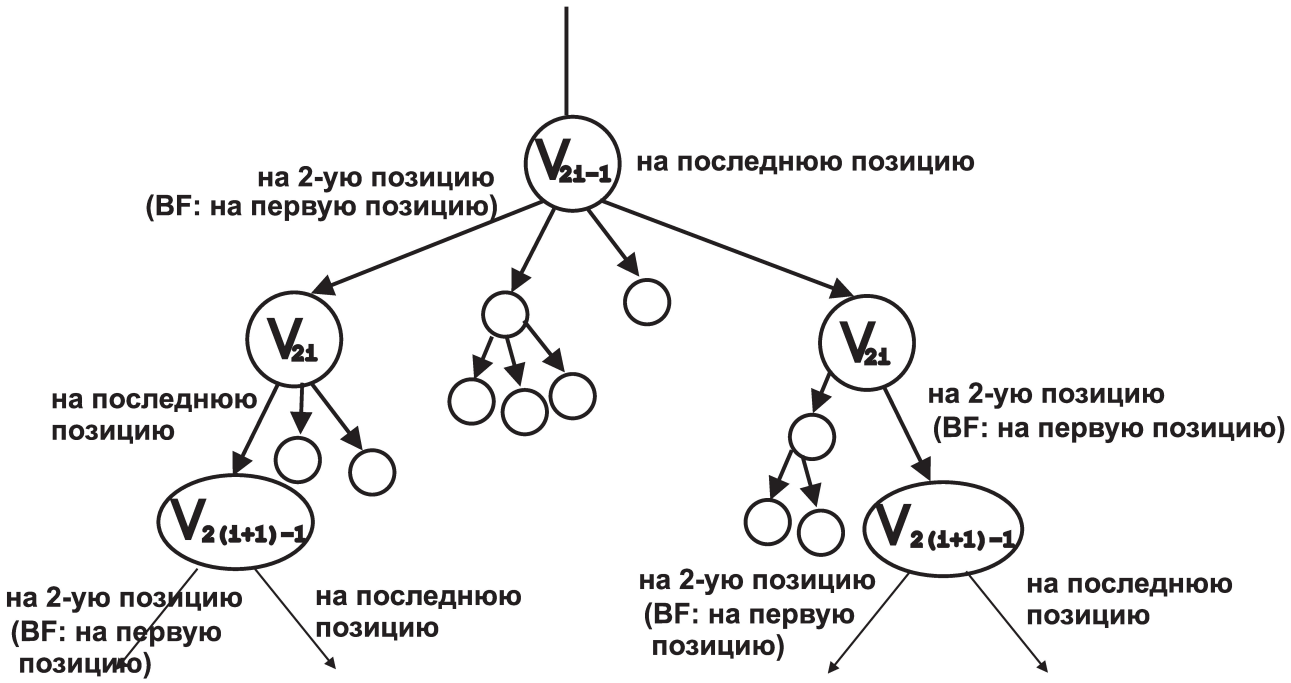

Рис. 5.8: Дерево поиска

Определение 5.2 *Канонические примеры, для которых выполняется*

$$\frac{1}{2}\delta - \sum_{j:=1}^{i-1} \delta_j \geq \delta_i,\ 2 \leq i \leq (n-1),$$

будем называть примерами случая SD (short delta).

Для случая *SD* выполняется

$$\delta_i > \frac{\sum\limits_{j:=1}^{i-1} \delta_j - \frac{1}{2}\delta}{2(n-i+1)},\quad 2 \leq i \leq (n-1),$$

т.к. $\frac{1}{2}\delta - \sum\limits_{j:=1}^{i-1} \delta_j \geq \delta_i > 0$, то $\sum\limits_{j:=1}^{i-1} \delta_j - \frac{1}{2}\delta < 0$. Например, случаю *SD* удовлетворяют примеры, для которых

$$\delta_i > 2\sum_{j:=1}^{i-1} \delta_j,\quad 2 \leq i \leq n.$$

Требования N пронумеруем в порядке EDD (early due date):
$(V_1, V_2, W_1, \ldots, V_{2i-1}, V_{2i}, W_i, \ldots, W_n, W_{n+1})$.

На рис. 5.8 представлено дерево поиска оптимального расписания, построенного алгоритмом A для *канонических DL-примеров* с использованием перебора подходящих позиций.

Определение 5.3 *Остов дерева поиска, получаемый "двоичным ветвлением" (рис. 5.8) будем называть "основным деревом".*

Лемма 5.22 *При использовании только правил исключения 1–3, то дерево поиска содержит "двоичное ветвление" (рис. 5.8). Ветвление происходит при выборе места для очередного требования V_{2i-1}. Для требования V_{2i} допустимой становится "противоположная" позиция, $i = 1, 2, \ldots, n$.*

Доказательство. Доказательство проведем методом математической индукции.

1. Покажем, что "двоичное ветвление" имеет место при выборе позиций для требований V_1 и V_2. Имеем множество N, состоящее из $3n+1$ требований, $N = \{V_1, V_2, W_1, \ldots, V_{2i-1}, V_{2i}, W_i, \ldots, W_n, W_{n+1}\}$. Требование $j^* = V_1$, момент начала обслуживания требований $t' = 0$.

   а. Покажем, что позиция 1 для требования V_1 является подходящей. Очевидно, что правило **б**) из теоремы 5.9 выполняется. Правило **а**) также выполняется, т.к.

   $$t' + p_{V_1} = 0 + a_1 < d_{V_2} = \frac{1}{2}\delta + a_2 + 2(n-1+1)(a_1 - a_2),$$

   т.е. позиция 1 для требования V_1 подходящая.

   Покажем, что при постановке V_1 на позицию 1 для требования V_2 (которое на второй итерации является j^*) позиция $(3n+1)$ будет подходящей. Правило **б**) выполняется. Легко убедиться, что

   $$t' + \sum_{j=1}^{3n} p_j = \sum_{j=1}^{2n} a_j + (n+1)b >$$

   $$> (n+1)b + \frac{1}{2}\delta + a_2 + a_4 + \ldots + a_{2n} + a_2.$$

   $$(n+1)b + \frac{1}{2}\delta + a_2 + a_4 + \ldots + a_{2n} + a_2 =$$

   $$= d_{W_{n+1}} + a_2 = \max_{j \in N'}\{d_j\} + \max_{j \in N'}\{p_j\} > d_j + p_j, \forall j \in N'.$$

   Правило **а**), очевидно, выполняется, т.к. позиция $3n+1$ последняя.

   б. Покажем, что позиция $3n+1$ для требования V_1 подходящая. Правило **а**), очевидно, выполняется, т.к. позиция $3n+1$ последняя. Покажем выполнение правила **б**). Видно, что

   $$0 + \sum_{j=1}^{3n+1} p_j = \sum_{j=1}^{2n} a_j + (n+1)b >$$

$$> (n+1)b + \frac{1}{2}\delta + a_2 + a_4 + \ldots + a_{2n} + a_1.$$

$$(n+1)b + \frac{1}{2}\delta + a_2 + a_4 + \ldots + a_{2n} + a_1 = d_{W_{n+1}} + a_1 =$$

$$= \max_{j \in N}\{d_j\} + \max_{j \in N}\{p_j\} > d_j + p_j, \forall j \in N.$$

Покажем, что если требование V_1, обслуживается $3n+1$ (последним) по порядку, то для требования V_2 становится подходящей только позиция 1. Очевидно, что правило **б**) выполняется. Правило **а**) также выполняется, т.к.

$$t' + p_{V_2} = 0 + a_2 < d_{W_1} = b + a_2.$$

2. Предположим, что "двоичное ветвление" имело место в группах $1, 2, \ldots, i-1$. В основном дереве первое требование из каждой пары занимает "вторую" (после соответствующего требования W) или "последнюю" позицию, а второе требование из пары – "противоположную" позицию.

3. В предположении п. 2 имеет место неравенство:

$$a_2 + a_4 + \ldots + a_{2i-2} + (i-2)b \le t' \le a_1 + a_3 + a_{2i-3} + (i-2)b,$$

где t' – момент времени, в котором "принимается решение" относительно i-ой группы требований $\{W_{i-1}, V_{2i-1}, V_{2i}\}$. Обозначим через $\bar{\delta}$ величину $\bar{\delta} = t' - (a_2 + a_4 + \ldots + a_{2i-2} + (i-2)b)$, для которого выполняется $0 \le \bar{\delta} \le \sum_{j=1}^{i-1} \delta_j < \delta$. Тогда

$$t' = a_2 + a_4 + \ldots + a_{2i-2} + (i-2)b + \bar{\delta}.$$

Имеется множество требований $N' = \{W_{i-1}, V_{2i-1}, V_{2i}, W_i, \ldots, W_n, W_{n+1}\}$, для которого требование $j^* = V_{2i-1}$. Учитывая, что $d_{W_{i-1}} < d_j$, $p_{W_{i-1}} \le p_j$, $\forall j \in N' \setminus \{W_{i-1}\}$, то при любом оптимальном расписании π выполняется $(W_{i-1} \to j)_\pi$, $\forall j \in N' \setminus \{W_{i-1}\}$. "Вторая" и "последняя" позиции для требования V_{2i-1} являются подходящими:

   а. Покажем, что текущая "вторая" позиция для требования V_{2i-1} подходящая. Правило **б**) выполняется, т.к. $t' + p_{W_{i-1}} + p_{V_{2i-1}} =$
   $= t' + b + a_{2i-1} = a_2 + a_4 + \ldots + a_{2i-2} + (i-2)b + \bar{\delta} + a_{2i-1} +$
   $+ b > d_{W_{i-1}} + p_{W_{i-1}} = a_2 + a_4 + \ldots + a_{2i-2} + (i-2)b + b.$
   Покажем, что выполняется и правило **а**).

Для случая SD выполняется: $t' + b + a_{2i-1} = a_2 + a_4 + \ldots + a_{2i-2} + (i-1)b + \bar{\delta} + a_{2i-1} < (i-1)b + \frac{1}{2}\delta + (a_2 + a_4 + \ldots + a_{2i-2} + a_{2i}) + 2(n - i + 1)(a_{2i-1} - a_{2i}) = d_{V_{2i}}$.

Покажем, что при постановке V_{2i-1} во "вторую" (после W_{i-1}) позицию для требования V_{2i} становится подходящей "последняя" позиция в списке $N'' = \{V_{2i}, W_i, \ldots, W_n, W_{n+1}\}$. Правило **б**) также выполняется: $i = 1, 2, 3$;

$$t' + b + a_{2i-1} + \sum_{j \in N''} p_j =$$

$$= a_2 + a_4 + \ldots + a_{2i-2} + (i-1)b + \bar{\delta} + (n - i + 2)b + \sum_{j:=2i-1}^{2n} a_j >$$

$$> a_2 + a_4 + \ldots + a_{2i-2} + a_{2i} + a_{2i+2} + \ldots + a_{2n} + (n+1)b + a_{2i},$$

поэтому

$$a_2 + a_4 + \ldots + a_{2i-2} + a_{2i} + a_{2i+2} + \ldots + a_{2n} + +(n+1)b + a_{2i} =$$
$$= d_{W_{n+1}} + a_{2i} = \max_{j \in N''} d_j + \max_{j \in N''} p_j > d_j + p_j, \forall j \in N''.$$

Правило **а**) для "последней" позиции, очевидно, выполняется, т.к. директивный срок "фиктивного требования" с директивным сроком $d_{|N''|+1} = +\infty > t' + b + a_{2i-1} + \sum_{j \in N''} p_j$.

б. Покажем, что текущая "последняя" позиция для требования V_{2i-1} является также подходящей. Правило **а**), очевидно, выполняется, т.к. имеем $d_{|N'|+1} = +\infty > t' + \sum_{j \in N'} p_j$. Покажем выполнение правила **б**), т.к.

$t' + \sum_{j \in N'} p_j = a_2 + a_4 + \ldots + a_{2i-2} + (i-1)b + \bar{\delta} +$

$+ (n - i + 2)b + \sum_{j:=2i}^{2n} a_j > a_2 + a_4 + \ldots + a_{2i-2} +$

$+ a_{2i} + a_{2i+2} + \ldots + a_{2n} + (n+1)b + a_{2i-1}$, поэтому

$a_2 + a_4 + \ldots + a_{2i-2} + a_{2i} + a_{2i+2} + \ldots + a_{2n} + (n+1)b +$

$+ a_{2i-1} = d_{W_{n+1}} + a_{2i-1} = \max_{j \in N'} d_j + \max_{j \in N'} p_j >$

$> d_j + p_j, \forall j \in N'$, т.е. правило **б**) выполняется.

Покажем, что при постановке V_{2i-1} на "последнюю" позицию для требования V_{2i} становится подходящей "вторая" позиция в списке $N'' = \{W_{i-1}, V_{2i}, W_i, \ldots, W_n, W_{n+1}\}$.

Правило **б**) выполняется, т.к. $t' + b + a_{2i} = a_2 + a_4 + \ldots + a_{2i-2} + (i-2)b + \bar{\delta} + b + a_{2i} > d_{W_{i-1}} + p_{W_{i-1}} = a_2 + a_4 + \ldots + a_{2i-2} + (i-2)b + b$.

Покажем, что выполняется и правило **а**):

$t' + b + a_{2i} = a_2 + a_4 + \ldots + a_{2i-2} + (i-1)b + \bar{\delta} + a_{2i} <$
$< ib + (a_2 + a_4 + \ldots + a_{2i-2} + a_{2i}) = d_{W_i}$, т. к. верно $b \gg \delta > \bar{\delta}$. $\square$

Лемма 5.23 *Правило исключения 4 не сокращает "двоичное ветвление" при выборе позиции для очередного требования V_{2i-1} в случае SD.*

Доказательство. Доказательство проводится аналогично доказательству леммы 5.22. Пусть "двоичное ветвление" имело место начиная с первой группы до группы $i-1$. Тогда выполняется

$$t' = a_2 + a_4 + \ldots + a_{2i-2} + (i-2)b + \bar{\delta}, \quad 0 \le \bar{\delta} \le \sum_{j=1}^{i-1} \delta_j < \delta.$$

1. Покажем, что правило исключения 4 не исключает из списка подходящих позиций для требования V_{2i-1} "вторую" позицию в списке $N' = \{W_{i-1}, V_{2i-1}, V_{2i}, W_i, \ldots, W_n, W_{n+1}\}$. (Первую позицию займёт требование W_{i-1}).

   Пусть $\pi_1 = (W_{i-1}, V_{2i-1}, V_{2i}, W_i, \ldots, W_n, W_{n+1})$,
   $\pi_2 = (W_{i-1}, V_{2i}, V_{2i-1}, W_i, \ldots, W_n, W_{n+1})$.

   Покажем, что $F(\pi_2) - F(\pi_1) > 0$.

   $F(\pi_2) - F(\pi_1) = T_{V_{2i}}(\pi) + T_{V_{2i-1}}(\pi) - T_{V_{2i-1}}(\pi') + T_{V_{2i}}(\pi') =$
   $= \max\{t' + b + a_{2i} - d_{a_{2i}},\ 0\} + \max\{t' + b + a_{2i} + a_{2i-1} - d_{V_{2i-1}},\ 0\} -$
   $- \max\{t' + b + a_{2i-1} - d_{V_{2i-1}},\ 0\} - \max\{t' + b + a_{2i-1} + a_{2i} - d_{a_{2i}},\ 0\}$.

   Очевидно, что $\max\{t' + b + a_{2i} + a_{2i-1} - d_{V_{2i-1}},\ 0\} -$
   $- \max\{t' + b + a_{2i-1} + a_{2i} - d_{V_{2i}},\ 0\} = 2(n-i+1)(a_{2i-1} - a_{2i})$ (из условия *канонического DL-примера*).

   a. Пусть $t' + b + a_{2i} - d_{V_{2i}} > 0$, то $t' + b + a_{2i-1} - d_{V_{2i-1}} > 0$ (из условия *канонического DL-примера*).

      Тогда $F(\pi_2) - F(\pi_1) = 2(n-i+1)(a_{2i-1} - a_{2i}) + (a_{2i} -$
      $- a_{2i-1} - 2(n-i+1)(a_{2i-1} - a_{2i})) < 0$. Правило исключения 4 сокращает список "подходящих" позиций, но при этом выполняется
      $t' + b + a_{2i} - d_{V_{2i}} > 0 \Rightarrow a_2 + a_4 + \ldots + a_{2i-2} + a_{2i} +$
      $+ (i-1)b + \bar{\delta} > (i-1)b + (a_2 + a_4 + \ldots + a_{2i-2} + a_{2i}) +$
      $+ \frac{1}{2}\delta + 2(n-i+1)(a_{2i-1} - a_{2i})$, поэтому верно
      $\bar{\delta} > \frac{1}{2}\delta + 2(n-i+1)(a_{2i-1} - a_{2i})$. Для случая SD это не выполняется, т.к. $\bar{\delta} \le \sum_{j=1}^{i-1} \delta_j < \frac{1}{2}\delta$.

б. $t' + b + a_{2i} - d_{V_{2i}} \leq 0$ и $t' + b + a_{2i-1} - d_{V_{2i-1}} > 0$.
$t' + b + a_{2i} - d_{V_{2i}} \leq 0 \Rightarrow a_2 + a_4 + \ldots + a_{2i-2} + a_{2i} + (i-1)b + \bar{\delta} \leq (i-1)b + (a_2 + a_4 + \ldots + a_{2i-2} + a_{2i}) + \frac{1}{2}\delta + 2(n-i+1)(a_{2i-1} - a_{2i}) \Rightarrow \bar{\delta} \leq \frac{1}{2}\delta + 2(n-i+1)(a_{2i-1} - a_{2i})$.
$F(\pi_2) - F(\pi_1) = 2(n-i+1)(a_{2i-1} - a_{2i}) - (\bar{\delta} - \frac{1}{2}\delta + (a_{2i-1} - a_{2i})) \geq 2(n-i+1)(a_{2i-1} - a_{2i}) - (\frac{1}{2}\delta + 2(n-i+1)(a_{2i-1} - a_{2i}) - \frac{1}{2}\delta + (a_{2i-1} - a_{2i})) = -(a_{2i-1} - a_{2i}) < 0$. То есть правило исключения 4 сокращает список "подходящих" позиций, но имеет место:
$t' + b + a_{2i-1} - d_{V_{2i-1}} > 0 \Rightarrow a_2 + a_4 + \ldots + a_{2i-2} + a_{2i-1} + (i-1)b + \bar{\delta} > (i-1)b + (a_2 + a_4 + \ldots + a_{2i-2} + a_{2i}) + \frac{1}{2}\delta \Rightarrow \bar{\delta} > \frac{1}{2}\delta + (a_{2i} - a_{2i-1})$.
Для случая SD это не выполняется, т.к. $\bar{\delta} \leq \sum_{j=1}^{i-1} \delta_j < \frac{1}{2}\delta$.

в. Пусть $t' + a_{2i} - d_{V_{2i}} < 0$ и $t' + a_{2i-1} - d_{V_{2i-1}} < 0$. Будем иметь $F(\pi_2) - F(\pi_1) = 2(n-i+1)(a_{2i-1} - a_{2i}) > 0$.

2. Правило исключения 4 не сокращает из списка подходящих позиций для требования V_{2i-1} "последнюю" позицию для требований подмножества $N' = \{W_{i-1}, V_{2i-1}, V_{2i}, W_i, \ldots, W_n, W_{n+1}\}$.

Рассмотрим частичное расписание требований множества N' $\pi = (W_{i-1}, V_{2i}, W_i, \ldots, W_n, W_{n+1}, V_{2i-1})$, обслуживание которого начинается с момента времени t'. Очевидно, что $d_{V_{2i}} \leq t' + p_{W_{i-1}} + p_{V_{2i}}$, $d_{W_i} \leq t' + p_{W_{i-1}} + p_{V_{2i}} + p_{W_i}$, $d_{W_{i-1}} \leq t' + p_{W_{i-1}}$. Все остальные требования из π запаздывают (см. условие *канонического DL-примера*).

Обозначим расписание $\pi = (W_{i-1}, \pi_1, \pi_2, V_{2i-1})$. Рассмотрим расписание $\pi' = (W_{i-1}, \pi_1, V_{2i-1}, \pi_2,)$. Очевидно, что в расписаниях π и π' требования из π_2 и требование V_{2i-1} запаздывают. При расписании π' суммарное запаздывание требований из π_2 увеличилось на $|\{\pi_2\}|a_{2i-1}$, а запаздывание требования V_{2i-1} сократилось на $P(\pi_2)$.

$F(\pi') - F(\pi) = |\{\pi_2\}|a_{2i-1} - P(\pi_2) > 0$, т.к. выполняется $a_{2i-1} > p_j, j \in \{\pi_2\}$. Таким образом $F(\pi) < F(j^*, i), \quad j^* \leq i \leq k$. □

Лемма 5.24 *Правило исключения 4 удаляет из списка "подходящих" позиций для очередного требования $V_{2i-1}, i = 1, 2, \ldots, n$, все позиции кроме текущих "второй" и "последней".*

Доказательство. Пусть

$$t' = a_2 + a_4 + \ldots + a_{2i-2} + (i-2)b + \bar{\delta}, \quad 0 \leq \bar{\delta} \leq \sum_{j=1}^{i-1} \delta_j < \delta.$$

Имеем множество требований

$$N' = \{W_{i-1}, V_{2i-1}, V_{2i}, W_i, \ldots, W_n, W_{n+1}\}.$$

Покажем, что $F(V_{2i-1}, k) > F(V_{2i-1}, k+1)$, где k – не "вторая" и не "последняя" позиция.

Рассмотрим 2 расписания, построенных с момента времени t':
$\pi_1 = (W_{i-1}, V_{2i}, W_i, V_{2(i+1)-1}, V_{2(i+1)}, \ldots, V_{2i-1}, X, \ldots, W_{n+1})$ и
$\pi_2 = (W_{i-1}, V_{2i}, W_i, V_{2(i+1)-1}, V_{2(i+1)}, \ldots, X, V_{2i-1}, \ldots, W_{n+1})$.

Не трудно показать, что все требования из π_1 и π_2 запаздывают (кроме, быть может, W_{i-1}, V_{2i} и W_i).

Тогда $F(\pi_1) > F(\pi_2)$, т.к. $p_{V_{2i-1}} > p_X$ для любого требования $X \in N' \backslash V_{2i-1}$. □

Аналогичные рассуждения можно провести для требования V_{2i}. Следовательно, при помощи алгоритма, использующего правило исключения 4, строятся только канонические расписания.

Лемма 5.25 *При использовании значений E_j, L_j для примеров случая SD "двоичное ветвление" в дереве поиска не сокращается.*

Доказательство. Для списка требований N, при использовании правила Эммонса выполняется только $(W_{i-1} \to j)_\pi \ \forall j \in \{V_{2i-1}, V_{2i}, W_i, \ldots, W_{n+1}\}$ при любом *каноническом DL-расписании*. Тогда $E_{V_{2i-1}} = (i-1)b + a_{2i-1}$, $E_{V_{2i}} = (i-1)b + a_{2i}$, $E_{W_i} = ib = d_{W_i}$. □

Лемма 5.26 *Для модифицированного примера двоичное ветвление в основном дереве сохранится.*

Доказательство. Из леммы 5.25 и вида *канонического DL-расписания*:

$E_{V_{2i-1}} = (i-1)b + a_{2i-1} < d_{V_{2i-1}}$;

$E_{V_{2i}} = (i-1)b + a_{2i} < d_{V_{2i}}$; $E_{W_i} = ib < d_{W_i}$,

т.е. $d'_i = d_i \forall i$, $\forall i \in N$. Модифицированный пример не отличается от исходного, т.к. $p'_i = p_i$, $d'_i = \max\{E_i, d_i\} = d_i$, $\forall i \in N$. □

Величины b_i не влияют на ветвления основного дерева, влияют только δ_i, $i = 1, 2, \ldots, n$. Американские учёные Ду и Леунг (J. Du & J. Y.-T. Leung) показали, что если ответ соответствующего примера **ЧНР** "ДА", то количество оптимальных расписаний чётно [114].

5.6.2 Трудоёмкость алгоритмов решения канонических DL-примеров

Теорема 5.11 *Для случая канонических* SD*-примеров алгоритмы, использующие только следующие способы сокращения перебора: правила исключения 1–4, использование E_j и L_j, построение модифицированного примера, имеют трудоёмкость не меньше $O(n2^{\frac{n-1}{3}-1})$ операций.*

Доказательство. В лемме 5.22 показано, что для очередного требования $V_{2i-1},\ i = 1, 2, \ldots, n-1$, "подходящими" являются две позиции.

Правило исключения 4 не сокращает этот список из двух позиций (лемма 5.23). В этом случае рассматривается две подзадачи, когда требование V_{2i-1} занимает текущую "вторую" или текущую "последнюю" позицию.

Процедура поиска подходящих позиций для очередного требования с максимальной продолжительностью обслуживания в каждой подзадаче имеет трудоёмкость $O(n)$ операций. Двоичное ветвление (рис. 5.8) выполняется для каждого требования $V_{2i-1},\ i = 1, 2, \ldots, n-1$, количество которых $(n-1)/3-1$. Следовательно трудоёмкость алгоритмов составляет не меньше $O(n2^{\frac{n-1}{3}-1})$ операций. □

Заметим, если пример не удовлетворяет случаю *SD*, то основное дерево не будет полным. Ветвления в дереве поиска будут продолжаться до тех пор пока выполняется неравенство $\bar{\delta} < \frac{1}{2}\delta + 2(n-i+1)(a_{2i-1} - a_{2i})$. Если же величина разности $a_{2k-1} - a_{2k} \approx a_{2l-1} - a_{2l}$, когда индексы чисел $k, l = 1, \ldots, n, k \neq l$, то сохранится, по крайней мере, "половина дерева" (ветвление имеет место для требований $i = 1, \ldots, \lceil n/2 \rceil$). Таким образом, экспоненциальная трудоёмкость алгоритма сохранится.

Стоит отметить, что для модифицированного примера **ЧНР**, для которого имеет место $\delta_1 \geq \ldots \geq \delta_n$, трудоёмкость алгоритма, использующего правила исключения 1–4, будет наименьшей.

5.6.3 Трудоёмкость алгоритмов решения задач случая BF

Примеры, удовлетворяющие следующим ограничениям, будем называть примерами случая *F2*:

$$\begin{cases} p_1 \geq p_2 \geq \ldots \geq p_n, \\ d_1 \leq d_2 \leq \ldots \leq d_n, \\ d_n - d_1 \leq p_n, \\ n = 2k, \\ \sum\limits_{i:=1}^{k} p_i < d_j < \sum\limits_{i:=k}^{n} p_i, \ j = 1, 2, \ldots, n, \\ p_1 - p_n \ll p_n, \\ \sum\limits_{i:=k+j+1}^{n} (p_{2j-1} - p_i) > d_{k+j} - d_{2j}, j = 1, \ldots, (k-1), \\ \sum\limits_{i:=k+j+1}^{n} (p_{2j} - p_i) > d_{k+j} - d_{2j}, j = 1, \ldots, (k-1). \end{cases} \tag{5.30}$$

Обозначим $(1, 2, \ldots, n) = (V_1, V_2, \ldots, V_{2j-1}, V_{2j}, \ldots, V_n)$.

Не трудно убедиться, что при любом расписании для любого примера случая (5.30) запаздывать будет ровно k требований. Напомним, что для данного случая F существует полиноминальный алгоритм решения (алгоритм 5.5).

Лемма 5.27 *Для примеров случая (5.30) дерево поиска содержит "двоичные ветвления" (рис.5.8). Ветвление происходит при выборе места для очередного требования V_{2i-1}. Для требования V_{2i} допустимой становится "противоположная" позиция, $i = 1, \ldots, (k-1)$. Правила исключения 1–4 не сокращают "двоичные ветвления".*

Доказательство. Доказательство проводится аналогично доказательству леммы 5.22 и леммы 5.23.

Предположим, что двоичное ветвление (когда первое требование из пары занимает "первую" или "последнюю" позицию, а второе требование из пары – "противоположную" позицию) имело место в группах $1, 2, \ldots, i-1$, $i < k$. Рассматривается подмножество из $n - 2i + 2$ требований $N' \subseteq N$, $N' = \{V_{2i-1}, V_{2i}, \ldots, V_n\}$. Множество требований N' начинает обслуживаться с момента времени $t' = p_2 + p_4 + \ldots + p_{2i-2} + \bar{\delta}$, где величина $0 \leq \bar{\delta} \leq \sum\limits_{j=1}^{i-1}(p_{2j-1} - p_{2j}) < p_n$. Требование с максимальной продолжительностью обслуживания $j^* = V_{2i-1}$. Покажем, что "первая" и "последняя" позиции для требования V_{2i-1} подходящие:

а. Текущая "первая" позиция является для требования V_{2i-1} подходящей. Очевидно, что правило **б**) выполняется. Покажем, что выполняется и

правило **а**): $t'+p_{2i-1} = p_2+p_4+\ldots+p_{2i-2}+\bar{\delta}+p_{2i-1} < d_{2i}$. Требование V_{2i-1} при постановке на "первую" позицию не запаздывает. Можно показать, что не запаздывает и "следующее" требование V_{2i}.

Покажем, что правило исключения 4 не исключает "первую" позицию. Рассмотрев 2 расписания $\pi' = (\pi_1, V_{2i-1}, V_{2i}, \pi_2)$ и $\pi'' = (\pi_1, V_{2i}, V_{2i-1}, \pi_2)$, где $P(\pi_1) = t'$, легко убедиться, что $F(\pi') = F(\pi'')$, т.к. при обоих расписаниях требования V_{2i-1} и V_{2i} не запаздывают.

Проводя аналогичные рассуждения не трудно доказать, что при постановке требования V_{2i-1} на "последнюю" позицию, для требования V_{2i} допустимой становится "первая" позиция.

б. Покажем, что текущая "последняя" позиция для требования V_{2i-1} является подходящей.

Очевидно, что правило **а**) выполняется, т.к. директивный срок фиктивного требования $d_{|N'|+1} = +\infty > t' + \sum\limits_{j\in N'} p_j$.

Покажем выполнение правила **б**):

$$t' + \sum_{j\in N'} p_j = p_2 + p_4 + \ldots + p_{2i-2} + \bar{\delta} + \sum_{j:=2i-1}^{2n} p_j > \sum_{j:=k}^{n} p_j + p_{2i-1};$$

$$\sum_{i:=k}^{n} p_i + p_{2i-1} > \max_{j\in N'} d_j + \max_{j\in N'} p_j > d_j + p_j, \forall j \in N'.$$

Покажем, что правило исключения 4 не сокращает перебор.
Рассмотрим два частичных модифицированных EDD расписания $\pi' = (\pi_1, \pi_2, V_{2i-1})$ и $\pi'' = (\pi_1, V_{2i-1}, \pi_2)$, составленные из требований множества N', обслуживание которых начинается с момента времени t'. Рассмотрим 2 случая:

1. Пусть при расписании π'' требование V_{2i-1} запаздывает. Очевидно, что при расписаниях π' и π'' запаздывают одни и те же требования. Для случая **F** запаздывающие требования при оптимальном расписании упорядочены в порядке SPT. Тогда $F(\pi') < F(\pi'')$.
2. Пусть при расписании π'' требование V_{2i-1} не запаздывает.
   Представим расписания в виде: $\pi' = (\pi_1, \pi_{21}, \pi_{22}, V_{2i-1})$ и $\pi'' = (\pi_1, V_{2i-1}, \pi_{21}, \pi_{22})$. Требования множества $\{\pi_{22}\}$ запаздывают при обоих расписаниях. Требования множества $\{\pi_{21}\}$, $|\pi_{21}| > 0$, при расписании π' не запаздывают. Учитывая, что количество запаздывающих требований при любом полном расписании равно $n/2 = k$,

то не трудно вычислить $|\{\pi_{22}\}|$. К шагу i уже выбрано $i-1$ запаздывающих требований, расположенных "в конце" полного расписания. Тогда $|\{\pi_{22}\}| = k-(i-1)-1$.

Множество $\{\pi_{22}\} = \{V_{n-|\{\pi_{22}\}|+1} \ldots, V_n\} = \{V_{k+i+1} \ldots, V_n\}$. Тогда при расписании π'' стало запаздывать еще одно требование $V_{k+i} \in \{\pi_{21}\}$.

$F(\pi') - F(\pi'') = C_{2i-1} - d_{2i-1} - |\{\pi_{22}\}|p_{2i-1} - (C_{k+i} - d_{k+i}) = t' + \sum\limits_{i\in\{\pi_1\}\bigcup\{\pi_{21}\}} p_i + \sum\limits_{i\in\{\pi_{22}\}} p_i + p_{2i-1} - d_{2i-1} - |\{\pi_{22}\}|p_{2i-1} - (t' + \sum\limits_{i\in\{\pi_1\}\bigcup\{\pi_{21}\}} p_i + p_{2i-1} - d_{k+i}) = d_{k+i} - d_{2i-1} - \sum\limits_{j:=k+i+1}^{n} (p_{2i-1} - p_j)$.

Для случая (5.30) выполняется $d_{k+i} - d_{2i-1} < \sum\limits_{j:=k+i+1}^{n} (p_{2i-1} - p_j)$, то $F(\pi') - F(\pi'') < 0$. Тогда $F(V_{2i-1}, i) > F(V_{2i-1}, |N'|),\ 1 \le i < |N'|$. Правило исключения 4 не сокращает перебор.

Аналогичные рассуждения можно провести для "последней" позиции требования V_{2i}, когда требование V_{2i-1} поставлено на "первую" позицию. □

Нетрудно показать, что $E_j = p_j$, $L_j = \sum\limits_{i:=1}^{n} p_i$. Тогда модифицированный пример, для которого $p'_j = p_j$, $d'_j = \max\{E_j = p_j, d_j\} = d_j$ совпадает с исходным примером. Значения E_j, L_j не сокращают "двоичное ветвление".

Теорема 5.12 *Для случая (5.30) алгоритмы, использующие только следующие правила исключения: правила исключения 1-4, использование E_j и L_j, построение модифицированного примера, имеют трудоёмкость $O(n2^{\frac{n}{2}})$ операций.*

Доказательство. Аналогично доказательству теоремы 5.11. □

Необходимо отметить, что для случая (5.30) алгоритм 5.5 находит оптимальное расписание за $O(n^2)$ операций.

В разделе 5.8 представлены экспериментальные исследования зависимости трудоёмкости решения примеров со случайно сгенерированными по различным законам распределения параметрами от размерности задачи n. Оказалось, что сгенерировать случайным образом пример, который будет иметь большую трудоёмкость решения с помощью алгоритма A, достаточно трудно. Результаты этого раздела (полиномиальная разрешимость частного

случая, для которого алгоритм A имеет экспоненциальную оценку трудоёмкости) демонстрируют актуальность разбиения общего случая задачи на совокупность частных случаев и получение для них свойств оптимальных расписаний.

5.7 Оценки SPT расписаний

Для целей данного раздела без ограничения общности будем предполагать, что требования множества N исходного примера I пронумерованы в порядке неубывания продолжительностей обслуживания

$$p_1 \leq p_2 \leq \ldots \leq p_n. \tag{5.31}$$

Через π_{spt} как и раньше будем обозначать SPT-расписание $(1, 2, \ldots, n)$.

Рассмотрим два примера $I' = \langle \{p_j, d'_j\}_{j \in N}, t_0 \rangle$ и $I'' = \langle \{p_j, d''_j\}_{j \in N}, t_0 \rangle$, где $d'_j \leq d''_j$, $j \in N$.

Лемма 5.28 *Для примеров I' и I'' выполняется $F^*(I') \geq F^*(I'')$.*

Доказательство. Пусть π' и π'' будут оптимальными расписаниями для примеров I' и I'', соответственно. Через $F'(\pi)$ и $F''(\pi)$ обозначим значение суммарного запаздывания при некотором расписании π, вычисленное для примеров I' и I'', соответственно.

Предположим, что выполняется $F^*(I') < F^*(I'')$. Вычислим значение суммарного запаздывания при расписании π' для примера I'':

$$F''(\pi') = \sum_{j=1}^{n} \max\{0, C_j(\pi') - d''_j\}.$$

Поскольку $d'_j \leq d''_j$, $j = 1, \ldots, n$, то для всех требований $j \in N$ выполняется

$$\max\{0, C_j(\pi') - d''_j\} \leq \max\{0, C_j(\pi') - d'_j\}.$$

.

Следовательно,

$$\sum_{j=1}^{n} \max\{0, C_j(\pi') - d''_j\} \leq \sum_{j=1}^{n} \max\{0, C_j(\pi') - d'_j\}. \tag{5.32}$$

Заметим, что в правой части неравенства (5.32) записано значение $F^*(I')$. Таким образом, выполняется $F''(\pi') \leq F^*(I') < F^*(I'')$. Это неравенство

противоречит предположению об оптимальности расписания π'' для примера I''. Следовательно, наше предположение не верно, и выполняется $F^*(I') \geq F^*(I'')$. Лемма доказана. $\square$

5.7.1 Первая оценка

Рассмотрим произвольный пример задачи $I = \langle \{p_j, d_j\}_{j \in N}, t_0 \rangle$ с оптимальным значением целевой функции $F^*(I)$. Обозначим $d_{\min} = \min\limits_{j \in N} d_j$ и $d_{\max} = \max\limits_{j \in N} d_j$.

Выберем величины C и δ такими, что

$$d_{\min} \leq C \leq d_{\max}, \tag{5.33}$$

$$\delta = \max\{C - d_{\max}, C - d_{\min}\}. \tag{5.34}$$

Пусть $S = t_0 + \sum\limits_{j=1}^{n} p_j$, и рассмотрим пример $I' = \left\langle \{p_j, d'_j = C\}_{j \in N},\ t_0 \right\rangle$. Справедлива теорема.

Теорема 5.13 *(первая оценка). Для выбранных значений C (5.33) и δ (5.34) верно неравенство*

$$|F^*(I) - F^*(I')| \leq n\delta \left(2 - 2\,\frac{C}{S} + \frac{\delta}{S}\right) + \delta.$$

Доказательство. Так как $d'_j = C$, $j \in N$, то SPT-расписание π_{spt} является оптимальным расписанием для примера I'.

Рассмотрим вспомогательные примеры $I'' = \left\langle \{p_j, d''_j = C - \delta\}_{j \in N}, t_0 \right\rangle$ и $I''' \left\langle \{p_j, d'''_j = C + \delta\}_{j \in N}, t_0 \right\rangle$. В тексте доказательства будем использовать запись $F' = F^*(I')$, $F'' = F^*(I'')$ и $F''' = F^*(I''')$. Оптимальным расписанием для примеров I'' и I''' также будет расписание π_{spt}.

В силу выбора значений C и δ имеют место неравенства $d'''_j \leq d_j \leq d''_j$ и $d'''_j \leq d'_j \leq d''_j$, $j \in N$. Согласно лемме 5.28, выполняется:

$$\begin{cases} F''' \leq F^* \leq F''; \\ F''' \leq F' \leq F''. \end{cases} \tag{5.35}$$

Определим множество $D_t(\pi) = \{j \in N : C_j(\pi) > t\}$. Для примеров I', I'', I''' множества $D_C(\pi_{spt}), D_{C-\delta}(\pi_{spt}), D_{C+\delta}(\pi_{spt})$ являются множествами запаздывающих требований при оптимальном расписании π_{spt}. Также, в силу $d'''_j \leq d'_j \leq d''_j, j \in N$, выполняется

$$D_{C+\delta}(\pi_{spt}) \subseteq D_C(\pi_{spt}) \subseteq D_{C-\delta}(\pi_{spt}). \tag{5.36}$$

Оценим величину $|D_C(\pi_{spt})|$. Поскольку имеет место $p_1 \le p_2 \le \ldots \le p_n$ и $\pi_{spt} = (1, 2, \ldots, n)$, то

$$|D_C(\pi_{spt})| = n - \left\lfloor n\frac{C}{S} \right\rfloor \le n - n\frac{C}{S} + 1,$$

где $\lfloor a \rfloor$ - целая часть числа a.

Аналогично,

$$|D_{C-\delta}(\pi_{spt})| \le n - n\frac{C-\delta}{S} + 1, \quad |D_{C+\delta}(\pi_{spt})| \le n - n\frac{C+\delta}{S} + 1.$$

При этом, т.к. $C_j(\pi_{spt}) \le S$, то справедливо:

$$\begin{aligned} F' &\le (n - n\frac{C}{S} + 1)(S - C) =: \tilde{F}'; \\ F'' &\le (n - n\frac{C}{S-\delta} + 1)(S - C) =: \tilde{F}''; \\ F''' &\le (n - n\frac{C}{S+\delta} + 1)(S - C) =: \tilde{F}'''. \end{aligned}$$

Легко проверяется, что

$$\tilde{F}'' - \tilde{F}' = \tilde{F}' - \tilde{F}''' = n\delta\left(2 - 2\frac{C}{S} + \frac{\delta}{S}\right) + \delta.$$

Также, из (5.36) следует неравенство

$$\tilde{F}''' - F''' \le \tilde{F}' - F' \le \tilde{F}'' - F''. \tag{5.37}$$

Рассмотрим следующие случаи.

1) Пусть $F' \le F^*$. Тогда, с учетом (5.35) и (5.37), выполняется

$$0 \le F^* - F' \le F'' - F' \le \tilde{F}'' - \tilde{F}' \le n\delta\left(2 - 2\frac{C}{S} + \frac{\delta}{S}\right) + \delta.$$

2) Пусть $F' > F^*$. Тогда, с учетом (5.35) и (5.37), выполняется

$$0 \le F' - F^* \le F' - F''' \le \tilde{F}' - \tilde{F}''' \le n\delta\left(2 - 2\frac{C}{S} + \frac{\delta}{S}\right) + \delta.$$

Следовательно мы имеем

$$|F^*(I) - F^*(I')| \le n\delta\left(2 - 2\frac{C}{S} + \frac{\delta}{S}\right) + \delta.$$

Теорема доказана. □

5.7.2 Вторая оценка

Рассмотрим функцию $f(t) = \sum\limits_{j=1}^{n} \max\{0, t_0 + \sum\limits_{i=1}^{j} p_i - t\}$. Учитывая начальную нумерацию требований (5.31), величина $t_0 + \sum\limits_{i=1}^{j} p_i$ для любого $j \in N$ равна моменту окончания требования j при SPT-расписании π_{spt}.

Построим пример $I(t) = \langle \{p_j, t\}_{j \in N}, t_0 \rangle$, при котором директивные сроки всех требований одновременно равны t. При этом, расписание π_{spt} будет оптимальным для примера $I(t)$ при любом значении t. Следовательно, значение $f(t)$ равно оптимальному значению суммарного запаздывания для примера $I(t)$, т.е. $f(t) = F^*(I(t))$.

Найдём вид функции $f(t)$. Через $\mathbb{T}_k$, $k = 1, 2, \ldots, n$, будем обозначать интервалы

$$\Big[t_0 + \sum_{j=1}^{k-1} p_j, t_0 + \sum_{j=1}^{k} p_j\Big).$$

Доопределим $\mathbb{T}_0 = (-\infty, t_0)$ и $\mathbb{T}_{n+1} = [t_0 + \sum\limits_{j=1}^{n} p_j, +\infty)$. Из регулярности критерия суммарного запаздывания следует, что $f(t)$ невозрастающая функция. Также, для $t \in \mathbb{T}_{n+1}$ выполняется $f(t) = 0$, поскольку в этом случае ни одно требование примера $I(t)$ не запаздывает при любом расписании π, т.е. $C_j(\pi) \leq t$.

Для $t \in \mathbb{T}_0 \bigcup \mathbb{T}_1$ при расписании π_{spt} запаздывают все требования примера $I(t)$, т.к. $t < t_0 + p_1 \leq C_j(\pi_{spt})$. Поэтому $f(t) = nt_0 + \sum\limits_{j=1}^{n} \sum\limits_{i=1}^{j} p_i - nt$. Для $t \in \mathbb{T}_k$, $k = 2, \ldots, n$, при расписании π_{spt} запаздывают только требования множества $\{k, k+1, \ldots, n\}$. Таким образом,

$$f(t) = (n-k+1)t_0 + \sum_{j=k}^{n} \sum_{i=1}^{j} p_i - (n-k+1)t, \quad t \in \mathbb{T}_k, k = 2, \ldots, n.$$

Распишем полученные выражения для $f(t)$:

$$\begin{aligned}
&t \in \mathbb{T}_1{:} f(t) = n(t_0 + p_1) + (n-1)p_2 + (n-2)p_3 \ldots + p_n - nt;\\
&t \in \mathbb{T}_2{:} f(t) = (n-1)(t_0 + p_1 + p_2) + (n-2)p_3 \ldots + p_n - (n-1)t;\\
&\quad \ldots\ldots\\
&t \in \mathbb{T}_n{:} f(t) = t_0 + p_1 + p_2 + p_3 + \ldots + p_n - t.
\end{aligned}$$

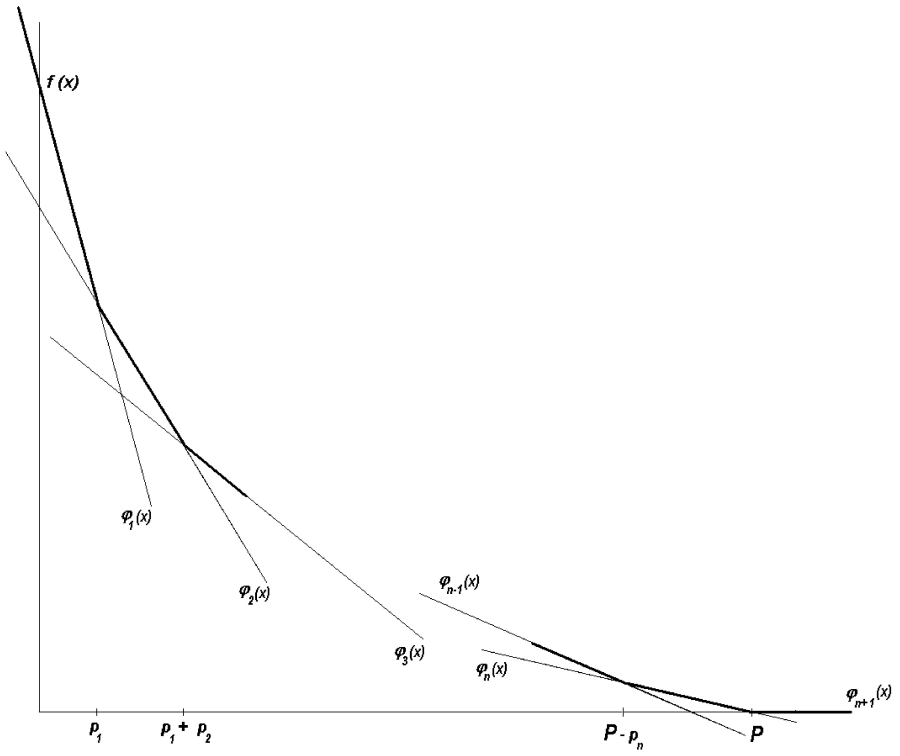

Рис. 5.9: Вид функции $f(x)$

Определим вспомогательные функции $\varphi_k(t)$, $k = 1, \ldots, n$:

$$\varphi_k(t) = (n-k+1)(t_0 + \sum_{j=1}^{k} p_j) + \sum_{j=k+1}^{n} (n-j+1)p_j - (n-k+1)t,$$

а также $\varphi_0(t) = \varphi_1(t)$ и $\varphi_{n+1}(t) = 0, \forall t$.

Тогда для $t \in \mathbb{T}_k$, $k = 0, 1, \ldots, n+1$, выполняется $f(t) = \varphi_k(t)$. Заметим, что для любых двух смежных интервалов $\mathbb{T}_{k-1}, \mathbb{T}_k$ верны неравенства:

$$\begin{aligned} \varphi_{k-1}(t) \leq \varphi_k(t), \quad & \text{для } t \in \mathbb{T}_k; \\ \varphi_k(t) \leq \varphi_{k-1}(t), \quad & \text{для } t \in \mathbb{T}_{k-1}. \end{aligned}$$

Следовательно, $f(t) = \max\limits_{k=0,1,\ldots,n+1} \varphi_k(t)$.

Вид функции $f(t)$ показан на рис. 5.9.

Лемма 5.29 $f(t)$ – *непрерывная функция.*

Доказательство. Так как внутри каждого интервала $\mathbb{T}_k$, $k = 1, 2, \ldots, n$, $f(x)$ является линейной функцией, то для доказательства леммы достаточно показать, что для точек $\tau_k = t_0 + \sum\limits_{j=1}^{k} p_j$, $k = 1, \ldots, n$, выполняется $\varphi_k(\tau_k) =$

$\varphi_{k+1}(\tau_k)$. Пусть $k \in \{1, \ldots, n-1\}$. Тогда

$$\varphi_k(\tau_k) = (n-k+1)(t_0 + \sum_{j=1}^{k} p_j) + \sum_{j=k+1}^{n} (n-j+1)p_j - (n-k+1)\tau_k =$$
$$= \sum_{j=k+1}^{n} (n-j+1)p_j$$

и

$$\varphi_{k+1}(\tau_k) = (n-k)(t_0 + \sum_{j=1}^{k+1} p_j) + \sum_{j=k+2}^{n} (n-j+1)p_j - (n-k)\tau_k =$$
$$= (n-k)p_{k+1} + \sum_{j=k+2}^{n} (n-j+1)p_j = \sum_{j=k+1}^{n} (n-j+1)p_j.$$

Пусть $k = n$. Тогда

$$\varphi_n(\tau_n) = t_0 + \sum_{j=1}^{n} p_j - (t_0 + \sum_{j=1}^{n} p_j) = 0 = \varphi_{n+1}(t).$$

Лемма доказана. □

Согласно лемме 5.28, выполняется $F^*(I(d_{\max})) \leq F^*(I) \leq F^*((d_{\min}))$. Следовательно, верно неравенство (*вторая оценка*):

$$f(d_{\max}) \leq F^*(I) \leq f(d_{\min}).$$

5.7.3 Задача построения множества требований с заданным оптимальным значением целевой функции

С помощью функции $f(t)$, определённой в предыдущем разделе, возможно решить задачу построения примера с заданным оптимальным значением целевой функции: по заданному значению $y^* \geq 0$ построить некоторый пример $I = \langle \{p_j, d_j\}_{j \in N}, t_0 \rangle$ такой, что $F^*(I) = y^*$.

Продолжительности обслуживания требований p_j и момент начала обслуживания t_0 берутся произвольно так, чтобы $p_1 \leq p_2 \leq \ldots \leq p_n$. Затем строится функция $f(t)$ и находится обратная к ней функция $f^{-1}(t)$ для точек $t \leq t_0 + \sum\limits_{j \in N} p_j$.

В этом случае $f(t) = \max\limits_{k=1,\ldots,n} \varphi_k(t)$ и $f^{-1}(y) = \max\limits_{k=1,\ldots,n} \varphi_k^{-1}(y)$. Все функции $\varphi_k(t)$ линейные, поэтому обратные к ним функции $\varphi_k^{-1}(y)$ находятся следующим образом

$$\varphi_k^{-1}(y) = \frac{(n-k+1)(t_0 + \sum\limits_{j=1}^{k} p_j) + \sum\limits_{j=k+1}^{n} (n-j+1)p_j}{(n-k+1)} + \frac{y}{(n-k+1)}.$$

Директивные сроки обслуживания требований примера I принимаются равными $d_j = f^{-1}(y^*)$, $\forall j \in N$. Для построенного примера I расписание π_{spt} будет оптимальным. Тогда выполняется $F^*(I) = f(f^{-1}(y^*)) = y^*$. Процедура построения примера I осуществляется за $O(n^2)$ операций.

5.8 Результаты экспериментальных исследований

В данном разделе представлены результаты экспериментальных исследований задачи минимизации суммарного запаздывания для одного прибора.

Исследование "средней" трудоёмкости алгоритма A.

В данном разделе приводятся результаты экспериментальных исследований "средней" трудоёмкости алгоритмов A и SBA на случайным образом сгенерированных параметров требований по различным законам распределения.

Для каждого $n = 25, 50, \ldots, 600$ и для каждого распределения генерировалось по 1000 примеров, а затем каждый построенный пример решался алгоритмами A и SBA, для которых подсчитывалось количество вершин (в дереве ветвлений для алгоритма A и списке подпримеров для алгоритма SBA). Поскольку обработка каждой вершины в обоих алгоритмах требует полиномиального количества операций, то выбранная для наблюдений величина (количество вершин в дереве поиска) является качественной характеристикой трудоёмкости решения примеров с помощью данных алгоритмов.

Каждый из параметров требований (продолжительности обслуживания p_j и директивные сроки d_j) генерировались с использованием следующих распределений: *нормальное* с параметрами $(0, 1)$, *равномерное* с параметрами $[-1; 1]$, *биномиальное* с параметрами $(1000, 0.5)$ и χ^2 с параметром 100.

Результаты экспериментов представлены на рис. 5.10. Для каждого значения n на графиках показано среднее значение трудоёмкости среди 1000 решённых примеров.

Как видно из полученных результатов средняя трудоёмкость решения примеров алгоритмами, использующими декомпозиционные свойства задачи, возрастает практически линейно с ростом n. Поэтому, одним из актуальных направлений исследований является нахождение трудоёмких для решения случаев, получение для них свойств оптимальных расписаний и построение соответствующих алгоритмов решения. На наш взгляд, это необходимо для понимания природы сложности NP-трудной проблемы $1||\sum T_j$.

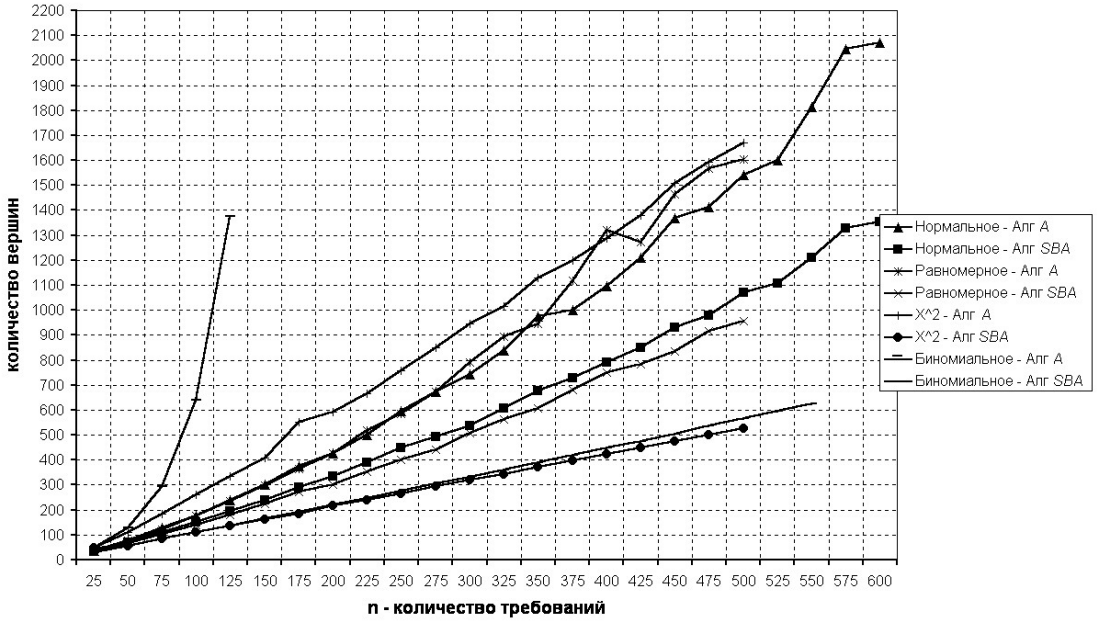

Рис. 5.10: Средняя трудоёмкость решения примеров с помощью алгоритмов A и SBA

Исследование эффективности приближённого алгоритма

Рассмотрим одну модификацию алгоритма A, с помощью которой за полиномиальное время можно строить приближённое решение примеров. Зафиксируем величину $\bar{H}$ – значение, которое будет ограничением на требуемую трудоёмкость нахождения приближённого решения.

Процесс работы нашего алгоритма разбит на два этапа. На первом этапе осуществляется построение дерева подпримеров алгоритма A "в ширину", пока количество вершин дерева не превысит величины $\bar{H}$. Затем для каждой из полученных концевых вершин неполного дерева подпримеров выполняем следующую процедуру: находим множество подходящих позиций для требования j^* и дальнейшее ветвление (т.е. декомпозицию на подпримеры) производим не для всех подходящих позиций, а только для случайно выбранной. Количество вершин в дереве ветвлений такого приближённого алгоритма не превышает $O(n\bar{H})$, обработка каждой вершины выполняется за $O(n)$ операций. Таким образом, нахождение приближённого решения описанным алгоритмом занимает $O(n^2\bar{H})$ операций.

Было проведено экспериментальное исследование эффективности приближённого алгоритма на классе тестовых примеров, описание которого приведено на стр. 221. Количество требований n в эксперименте варьировалось

Таблица 5.5: Эффективность приближённого алгоритма

n	$\Delta_{\min}(\%)$	$\Delta_{avg}(\%)$	$\Delta_{\max}(\%)$	$H_A(e_1)$
10	0	0	0	461
12	0	0	0	1,715
14	0	0	0	6,434
16	0	0	0	24,309
18	0	4.46	4.66	92,377
20	0	15.77	16.33	352,715
22	0	15.17	15.39	1,352,077
24	1.64	9.61	09.68	5,200,299
26	0.93	15.95	16.57	20,058,229
28	5.43	25.10	26.35	77,558,759
30	6.70	22.27	23.02	300,540,194

от 10 до 30. Для каждого значения n было сгенерировано по 1000 примеров. Пусть H_A обозначает количество вершин в дереве ветвлений (т.е. трудоёмкость) алгоритма A; $\Delta_{\min}$, Δ_{avg}, $\Delta_{\max}$ – соответственно, минимальное, среднее и максимальное значения относительной погрешности (в процентах) среди тысячи экспериментов для каждого n. Результаты экспериментальных исследований приведены в таблице 5.5. При проведении экспериментов значение $\bar{H}$ бралось равным n^3.

Исследование оценок оптимального значения целевой функции

Рассмотрим пространственное представление примеров задачи $1 \,||\, \sum T_j$, описание которого приведено в разделе 5.1. Для целей данного раздела без ограничения общности примем момент начала обслуживания t_0 любого примера равным нулю. Тогда пример задачи описывается точкой в $2n$-мерном пространстве. Далее в этом разделе понятие примера и соответствующей точки $2n$-мерного пространства будем отождествлять. Для получения оптимального решения все примеры решаются с помощью алгоритма A. Через $H_A(I)$ будем обозначать трудоёмкость решения примера I, т.е. величина $H_A(I)$ равна количеству вершин в дереве поиска алгоритма A.

Для получения оценок оптимального значения целевой функции используется следующая схема:

1. По исходному примеру I строится несколько примеров, трудоёмкость решения которых с помощью алгоритма A не превышает наперед заданной величины $\bar{H}$;

2. Для построенных примеров находятся оптимальные значения целевой функции, которые затем используются в качестве точек построения интерполяционного полинома, с помощью которого оценивается оптимальное значение целевой функции исходного примера I.

Точку, соответствующую исходному примеру I, будем обозначать через x_0. Проведем через точку x_0 в $2n$-мерном пространстве прямую, которая обладает следующим свойством: по обе стороны от точки x_0 на прямой должны существовать такие точки, трудоёмкость решения которых не более $\bar{H}$.

Рассматриваются три модели построения прямых (лучей), проходящих через точку x_0 и обладающих указанным свойством:

- **Мультипликативная модель (1).**
  Строится луч, точки которого $(p'_1, \ldots, p'_n, d'_1, \ldots, d'_n)$ задаются условиями $p'_j = p_j$, $d'_j = \frac{d_j}{k}$, $j = 1, \ldots, n$, $k \in (0, +\infty)$.

- **Мультипликативная модель (2).**
  Здесь строится луч, точки которого $(p'_1, \ldots, p'_n, d'_1, \ldots, d'_n)$ задаются уравнением $p'_j = p_j$, $d'_j = kd_j$, $j = 1, \ldots, n$, $k \in (0, +\infty)$.

- **Аддитивная модель.**
  Здесь строится прямая, точки которой $(p'_1, \ldots, p'_n, d'_1, \ldots, d'_n)$ задаются уравнением $p'_j = p_j$, $d'_j = d_j + k$, $j = 1, \ldots, n$, $k \in (-\infty, +\infty)$.

Для всех трех моделей выполняется условие на существование точек с трудоёмкостью решения не более $\bar{H}$. Это следует из того факта, что "слева" от точки x_0 ($k \to +\infty$ для первой модели, $k \to 0$ для второй модели и $k \to -\infty$ для третьей модели) будут существовать точки, у которых все директивные сроки будут меньше, чем минимальная продолжительность обслуживания $p_{\min}$. Для таких примеров оптимальное расписание строится за $O(n \log n)$ операций (SPT-расписание). Аналогичным образом, "справа" от точки x_0 ($k \to 0$ для первой модели, $k \to +\infty$ для второй модели и $k \to +\infty$ для третьей модели) будут существовать точки, у которых все директивные строки будут за пределами интервала планирования. Для таких примеров, любое из $n!$ расписаний будет оптимальным.

На рис. 5.11 приведены графики зависимости оптимального значения целевой функции и трудоёмкости решения примера для мультипликативной модели (2). Пример, для которого построены данные графики, принадлежит классу тестовых примеров, описанному на стр. 221 при $n = 10$.

Выберем такой отрезок значений k для всех трех моделей, что директивные сроки получаемых примеров принадлежат интервалу планирования $[0, \sum\limits_{j \in N} p_j]$. На полученном отрезке найдём l корней полинома Чебышёва $z_1, z_2, \ldots, z_l$. Выберем из них такие точки $y_1, y_2, \ldots, y_m$, что $H_A(y_i) \leq \bar{H}$ $(i = 1, \ldots, m)$[4]. Далее, по выбранным точкам $y_1, \ldots, y_m$ строится интерполяционный полином Лагранжа, с помощью которого оценивается оптимальное значение целевой функции в исходной точке x_0. Такая же схема может быть применена для получения оценки трудоёмкости решения примера с помощью алгоритма A.

На рис.5.12 приведены графики зависимости абсолютной погрешности интерполяции оптимального значения целевой функции и трудоёмкости решения от значения $\bar{H}$ для одного тестового примера со стр. 221 при $n = 10$.

Гипотеза. На приборе необходимо обслужить n требований
с $d_1 \leq d_2 \leq \ldots \leq d_n$ и $p_1 = \max\limits_{j=1,\ldots,n} p_j$. Пусть алгоритм A находит расписание за полиноминальное от n количество операций. Если $F(1, \pi_A(p_1)) < F(\pi_A(0), 1)$, тогда $F(1, \pi^*(p_1)) < F(\pi^*(0), 1)$, где $\pi_A(t)$ и $\pi^*(t)$ – расписания, построенные алгоритмом A и оптимальное с момента времени t, обслуживания множества требований {2,3,. . . , n}.

[4]Метод отбора точек по указанному условию заключается в следующем. Построим пример, соответствующий корню полинома Чебышёва, и начнём его решать с помощью алгоритма A. Как только количество вершин дерева ветвлений превышает значение $\bar{H}$, то процесс решения останавливается и точка считается не удовлетворяющей условию. Такая схема отбора позволяет найти не только требуемые точки, но также оптимальное значение целевой функции в них.

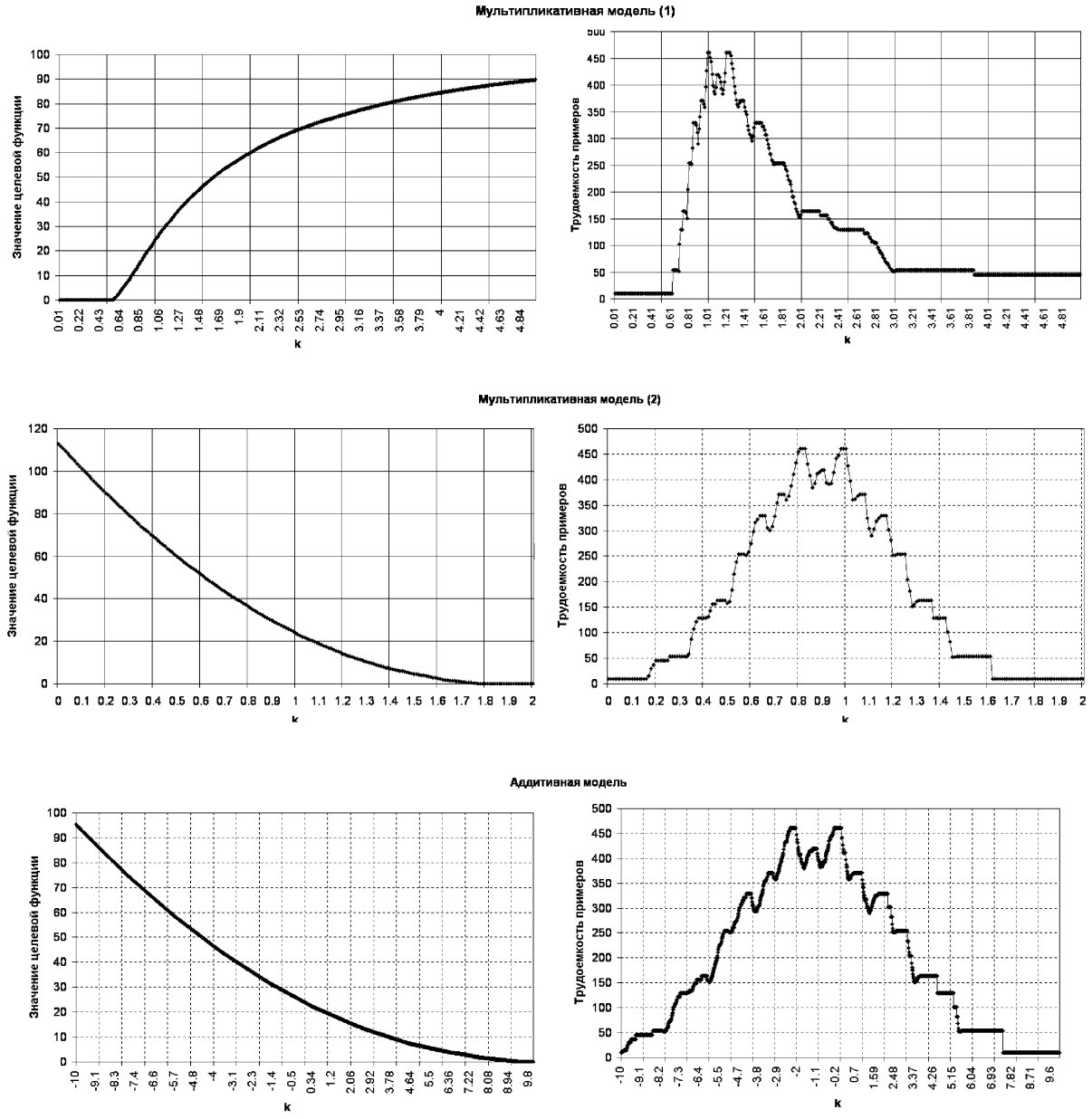

Рис. 5.11: Зависимость оптимального значения целевой функции и трудоёмкости решения от параметра k, $n = 10$.

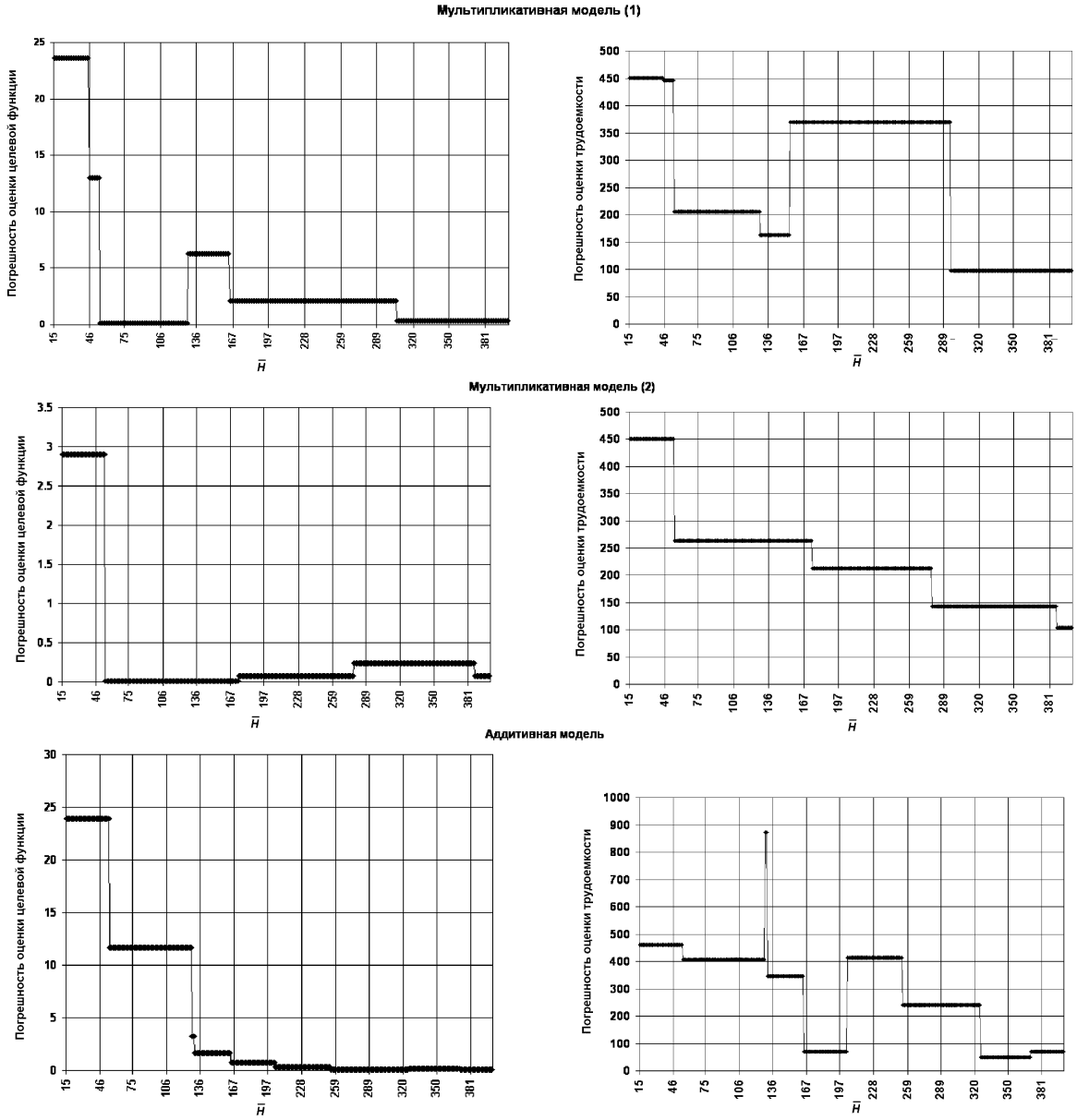

Рис. 5.12: Зависимость абсолютной погрешностей оценок оптимального значения целевой функции и трудоёмкости решения от значения $\bar{H}$ для тестового примера со стр. 221, $n = 10$.

Глава 6

Полиномиально и псевдополиномиально разрешимые случаи задачи $1 \mid\mid \sum T_j$

В данной главе рассматривается частный случай задачи минимизации суммарного запаздывания для одного прибора, когда параметры требований удовлетворяют условиям:

$$\begin{cases} p_1 \geq p_2 \geq \ldots \geq p_n; \\ d_1 \leq d_2 \leq \ldots \leq d_n. \end{cases} \tag{6.1}$$

Предлагается подход решения задачи в случае (6.1), исходное множество требований N разбивается на $\Bbbk$ непересекающихся подмножеств требований $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_\Bbbk$, таких, что $N = \mathcal{M}_1 \bigcup \mathcal{M}_2 \bigcup \ldots \bigcup \mathcal{M}_\Bbbk$ и величина $\max\limits_{i,j \in \mathcal{M}_\nu} |d_i - d_j| \leq \min\limits_{j \in \mathcal{M}_\nu} p_j$ для любого подмножества $\mathcal{M}_\nu$, где $\nu = 1, 2, \ldots, \Bbbk$, а затем на основе этого разбиения строится оптимальное расписание.

В данной главе будут использоваться следующие обозначения. Для нумерации подмножеств $\mathcal{M}$ будем использовать символ ν. Номер подмножества, которое содержит требование $j \in N$, будем обозначать через $\gamma(j)$; таким образом, по определению выполняется $j \in \mathcal{M}_{\gamma(j)}$.

В разделе 6.1 формулируются и доказываются свойства оптимальных расписаний в случае (6.1). На основе этих свойств в разделах 6.3.1-6.3.4 приводятся псевдополиномиальные и полиномиальные алгоритмы решения всех подслучаев случая (6.1): B-1, B-$\Bbbk$, C-1 и B-n. Описание подслучаев приведено в таблице 6.1. Рассматриваемые случаи схематично изображены на рис. 6.1.

6.1 Свойства оптимальных расписаний

В данном разделе доказываются две леммы, которые отражают свойства оптимальных расписаний в исследуемом случае (6.1).

Таблица 6.1: Алгоритмы и разрешимые случаи, представленные в главе 6.

Алгоритм	Разрешимый случай	Трудоёмкость
B-$\mathbb{k}$	$\begin{cases} p_1 \geq p_2 \geq \ldots \geq p_n \quad (*) \\ d_1 \leq d_2 \leq \ldots \leq d_n \quad (**) \end{cases}$	$\leq O(n^2 \sum p_j)$
B-1	$\begin{cases} (*), (**) \\ d_n - d_1 \leq p_n \end{cases}$	$O(n \sum p_j)$
C-1	$\begin{cases} (**) \\ d_n - d_1 \leq 1 \end{cases}$	$O(n^2)$
B-n	$d_j - d_{j-1} > p_j, \ j = 2, 3, \ldots, n$	$O(n^2)$

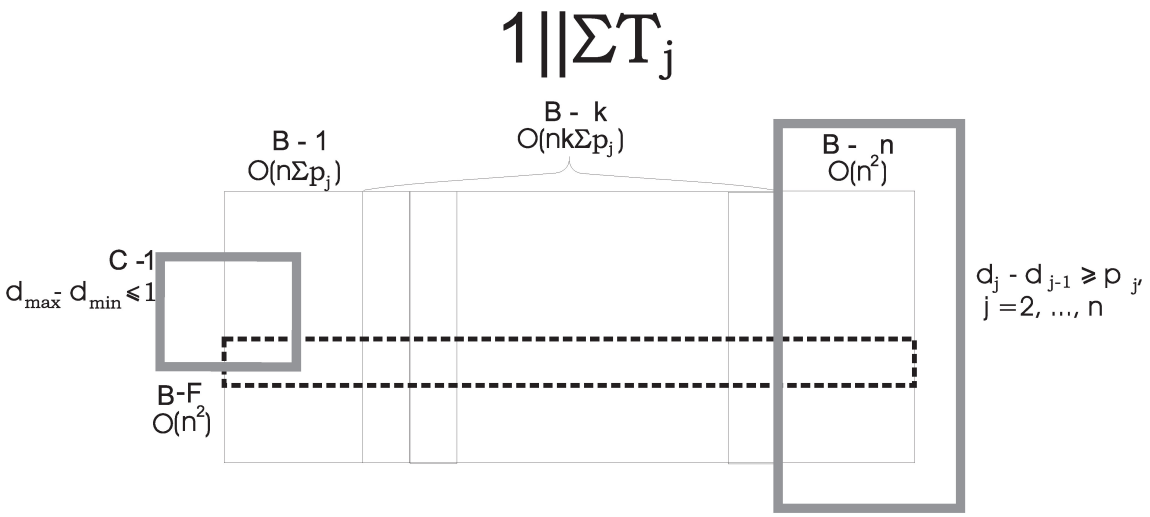

Рис. 6.1: Частные случаи задачи, исследуемые в главе 6.

Лемма 6.1 *В случае (6.1) существует оптимальное расписание π^*, при котором выполняется либо $(k \to i)_{\pi^*}$, либо $(j \to k)_{\pi^*}$ для любой тройки требований $i, j, k \in N$ такой, что $|d_i - d_j| \leq \min\{p_i, p_j\}$, $k < \min\{i, j\}$ и $(i \to j)_{\pi^*}$.*

Доказательство. Предположим, что при некотором оптимальном расписании π^* существуют такие требования $i, j, k \in N$, что $|d_i - d_j| \leq \min\{p_i, p_j\}$ и $k < \min\{i, j\}$ и выполняется $(i \to k \to j)_{\pi^*}$. То есть расписание π^* имеет вид $\pi^* = (\pi_1, i, \pi_2, k, \pi_3, j, \pi_4)$.

Построим расписания $\pi' = (\pi_1, \pi_2, k, i, \pi_3, j, \pi_4)$ и $\pi'' = (\pi_1, i, \pi_2, j, \pi_3, k, \pi_4)$. Покажем, что выполняется либо $F(\pi') \leq F(\pi^*)$, либо $F(\pi'') \leq F(\pi^*)$.

Поскольку $k < \min\{i, j\}$, то в силу (6.1) для требований i, j, k выполняется $p_k \geq p_i$, $p_k \geq p_j$, $d_k \leq d_i$ и $d_k \leq d_j$.

Рассмотрим следующие три случая.

а) Пусть $C_k(\pi^*) \leq d_k$. В этом случае, при расписании π' выполняется

$$C_i(\pi') = C_k(\pi^*) \leq d_k \leq d_i.$$

Следовательно, требования i и k не запаздывают при обоих расписаниях π^* и π'. Таким образом, в этом случае выполняется $F(\pi') \leq F(\pi^*)$.

b) Пусть $C_k(\pi^*) > d_k$ и $C_k(\pi^*) \leq d_j$. В этом случае, требование k запаздывает при расписании π^*, т.е. $T_k(\pi^*) = C_k(\pi^*) - d_k > 0$. Поскольку $\mid d_i - d_j \mid \leq \min\{p_i, p_j\} \leq p_k$ и $C_k(\pi^*) \leq d_j$, то выполняется

$$C_i(\pi^*) \leq C_k(\pi^*) - p_k \leq d_j - p_k \leq d_i.$$

Это означает, что требование i не запаздывает при расписании π^*. Тогда, в силу $C_i(\pi') = C_k(\pi^*)$ и $C_k(\pi') = C_k(\pi^*) - p_i$, выполняется

$$\begin{aligned} F(\pi') - F(\pi^*) \leq & \max\{0, C_k(\pi^*) - d_i\} - \max\{0, C_k(\pi^*) - p_i - d_k\} - \\ & -(C_k(\pi^*) - d_k) \leq 0. \end{aligned}$$

Таким образом, в этом случае выполняется $F(\pi') \leq F(\pi^*)$.

c) Пусть $C_k(\pi^*) > d_k$ и $C_k(\pi^*) > d_j$. В этом случае, требование k запаздывает при обоих расписаниях π^* и π''. Требование j запаздывает при расписании π^*, поскольку $d_j < C_k(\pi^*) < C_j(\pi^*)$. Также, выполняется $T_j(\pi'') = \max\{0, C_k(\pi^*) - p_k + p_j - d_j\}$. Следовательно,

$$\begin{aligned} F(\pi'') - F(\pi^*) \leq & \max\{0, C_k(\pi^*) - p_k + p_j - d_j\} + \\ & + C_j(\pi^*) - d_k - C_k(\pi^*) + d_k - C_j(\pi^*) + d_j \leq \\ & \leq \max\{0, C_k(\pi^*) - p_k + p_j - d_j\} - C_k(\pi^*) + d_j \leq 0. \end{aligned}$$

Таким образом, в этом случае выполняется $F(\pi'') \leq F(\pi)$.

В заключение отметим, что, если $F(\pi') = F(\pi^*)$ или $F(\pi'') = F(\pi^*)$, то расписание π' или π'' также является оптимальным. Если $F(\pi') < F(\pi^*)$ или $F(\pi'') < F(\pi^*)$, то получаем противоречие с предположением об оптимальности расписания π^*. Это означает, что не существует оптимального расписания π^*, при котором выполняется $(i \to k \to j)_{\pi^*}$, и любое оптимальное расписание удовлетворяет утверждению леммы. □

Лемма 6.2 *В случае (6.1) существует оптимальное расписание π^*, при котором выполняется либо $(k \to j)_{\pi^*}$, либо $((k+1) \to k)_{\pi^*}$ для любой пары требований $k, j \in N$ такой, что $k < j$.*

Доказательство. Идея данного доказательства аналогична идее доказательства леммы 6.1.

Предположим, что при некотором оптимальном расписании π^* существуют требования $i, j \in N$ такие, что $k < j$ и $(j \to k \to (k+1))_{\pi^*}$. То есть расписание π^* имеет вид $\pi^* = (\pi_1, j, \pi_2, k, \pi_3, k+1, \pi_4)$ Построим два расписания $\pi' = (\pi_1, \pi_2, k, j, \pi_3, k+1, \pi_4)$ и $\pi'' = (\pi_1, j, \pi_2, k+1, \pi_3, k, \pi_4)$. Покажем, что выполняется либо $F(\pi') \le F(\pi^*)$, либо $F(\pi'') \le F(\pi^*)$.

Поскольку $k < j$, то выполняется $p_k \ge p_{k+1} \ge p_j$ и $d_k \le d_{k+1} \le d_j$.

Рассмотрим следующие три случая.

a) Пусть $C_k(\pi^*) \le d_k$. В этом случае требования j и k не запаздывают при обоих расписаниях π^* и π'. Следовательно $F(\pi') \le F(\pi^*)$.

b) Пусть $C_k(\pi^*) > d_k$ и $C_k(\pi^*) \le d_{k+1}$. Поскольку $d_{k+1} \le d_j$, то выполняется $C_j(\pi') = C_k(\pi^*) \le d_{k+1} \le d_j$. Следовательно,

$$F(\pi') - F(\pi^*) \le \max\{0, C_k(\pi^*) - p_j - d_k\} - (C_k(\pi^*) - d_k) \le 0.$$

c) Пусть $C_k(\pi^*) > d_k$ и $C_k(\pi^*) > d_{k+1}$. Поскольку $p_k \ge p_{k+1}$, то выполняется $C_k(\pi'') = C_{k+1}(\pi^*)$ и $C_{k+1}(\pi'') \le C_k(\pi^*)$. Следовательно,

$$F(\pi'') - F(\pi^*) \le \max\{0, C_{k+1}(\pi'') - d_{k+1}\} + d_{k+1} - C_k(\pi^*) \le 0.$$

Если $F(\pi') = F(\pi^*)$ или $F(\pi'') = F(\pi^*)$, то расписание π' или π'' также является оптимальным. Если $F(\pi') < F(\pi^*)$ или $F(\pi'') < F(\pi^*)$, то получаем противоречие с предположением об оптимальности расписания π^*. Это означает, что не существует оптимального расписания π^*, при котором выполняется $(j \to k \to (k+1))_{\pi^*}$, и любое оптимальное расписание удовлетворяет утверждению леммы. □

6.2 Подход к решению задачи

В данном разделе приводится описание подхода к решению примеров, который позволяет построить псевдополиномиальный алгоритм, с помощью которого может быть построено оптимальное расписание для примеров в случае (6.1) за не более $O(n^2 \sum p_j)$ операций. Естественно, предполагается, что продолжительности обслуживания целые положительные числа.

Приведём описание *процедуры разбиения* исходного множества требований N на $\Bbbk$ непересекающихся подмножеств $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_\Bbbk$ таких, что $N = \mathcal{M}_1 \bigcup \mathcal{M}_2 \bigcup \ldots \bigcup \mathcal{M}_\Bbbk$ и $\max\limits_{i,j \in \mathcal{M}_\nu} |d_i - d_j| \le \min\limits_{j \in \mathcal{M}_\nu} p_j$.

Алгоритм 6.1 Процедура разбиения

1: $\Bbbk := 1$, $\alpha_1 := 1$;
2: **for all** $j = 2, 3, \ldots, n$ **do**
3: **if** $d_j - d_{\alpha_\Bbbk} > p_j$ **then**
4: $\beta_\Bbbk := j - 1$; $\Bbbk := \Bbbk + 1$; $\alpha_\Bbbk := j$;
5: **end if**
6: **end for**
7: $\beta_\Bbbk := n$; $\mathcal{M}_\nu = \{\alpha_\nu, \alpha_{\nu+1}, \ldots, \beta_\nu\}$, $\nu = 1, 2, \ldots, \Bbbk$.

Данная процедура однозначно разбивает исходное множество N на $\Bbbk$ подмножеств, $\Bbbk \leq n$, за $O(n)$ операций. В дальнейшем, для обозначения требований с наименьшим и наибольшим номером в множестве $\mathcal{M}_\nu$ будем использовать запись α_ν и β_ν, соответственно.

Например, для множества из трех требований $N = \{1, 2, 3\}$ с параметрами: $p_1 = 10, p_2 = 10, p_3 = 2$, $d_1 = 7, d_2 = 9, d_3 = 10$, будет получено разбиение на два подмножества $\mathcal{M}_1 = \{1, 2\}$ и $\mathcal{M}_2 = \{3\}$.

Согласно леммам 6.1 и 6.2, в случае (6.1) при построении оптимального расписания π^* можно исключить из рассмотрения все расписания π, при которых:

(a) существует требования $i, j, k \in N$ такие, что $k \leq \min\{i, j\}$ и $i, j \in M_\nu$, $\nu \geq \gamma(k)$, и выполняется $(i \to k \to j)_\pi$;
или

(b) существуют требования j, $k \in N$ такие, что $k < j$, и выполняется $(j \to k \to (k+1))_\pi$.

Для произвольного примера $I = \langle \{p_j, d_j\}_{j \in N}, t_0 \rangle$ определим *параметрический пример* $I_k(t) = \langle \{p_j, d_j(t)\}_{j \in N_k}, 0 \rangle$, где $N_k = \{k, k+1, \ldots, n\}$ и $d_j(t) = d_j - d_n + t - t_0$ для некоторого $k = 1, 2, \ldots, n$. Таким образом, параметрический пример $I_k(t)$ – это пример исследуемой задачи обслуживания требований множества N_k с моментом начала обслуживания равным нулю и директивными сроками $d_j(t)$, линейно зависящими от параметра t. Не ограничивая общности будем полагать, что $d_1 \leq d_2 \leq \ldots \leq d_n$.

Согласно (6.1), для любого $t \in \mathbb{R}$ выполняется $d_1(t) \leq d_2(t) \leq \ldots \leq d_n(t)$. При этом, исходный пример I равен параметрическому примеру $I_1(t)$ при $t = d_n$, т.е. $\Pi^*(I) = \Pi^*(I_1(d_n))$ (см. раздел 5.1).

Через $\pi_k^*(t)$ и $F_k^*(t)$ обозначим оптимальное расписание и оптимальное значение целевой функции для примера $I_k(t)$, соответственно.

Суть предлагаемого подхода к решению примеров в случае (6.1) заключается в построении оптимальных расписаний для примеров $I_k(t)$ в порядке

$k = n, n-1, \ldots, 1$ для $t \in [t_0; d_n]$. При этом, расписание $\pi_k^*(t)$ строится на основе ранее построенных расписаний $\pi_j^*(t')$, $j > k$, $t' \in [t_0; d_n]$.

Рассмотрим следующий пример. Пусть оптимальные расписания $\pi^*(t)$ построены для примеров $I_{k+1}(t)$ в каждой точке $t \in [t_0, d_n]$. Предположим, что $d_n - d_{k+1} \leq p_n$. Согласно лемме 6.1 в каждой точке t требование k может быть обслужено при оптимальном расписании $\pi_k^*(t)$ на первой или на последней позиции, т.е. до или после всех требований множества N_{k+1}. Таким образом, $\pi_k^*(t)$ является наилучшим из двух расписаний по величине суммарного запаздывания

$$(k, \pi_{k+1}^*(t - p_k)), \quad (\pi_{k+1}^*(t), k).$$

Заметим, что при постановке требования k на первую позицию в расписании $\pi_k^*(t)$ оптимальный порядок требований множества N_{k+1} с момента времени p_k соответствует порядку требований при расписании $\pi_{k+1}^*(t - p_k)$.

При этом, значение оптимального суммарного запаздывания $F_k^*(t)$ для примера $I_k(t)$ (т.е. значение целевой функции при расписании $\pi_k^*(t)$) может быть вычислено по формуле

$$F_k^*(t) = \min\Big\{\max\{0, p_k - d_k(t)\} + F_{k+1}^*(t - p_k),$$
$$F_{k+1}^*(t) + \max\Big\{0, \sum_{j=k}^{n} p_j - d_k(t)\Big\} \quad \Big\},$$

где первое слагаемое является значением суммарного запаздывания при расписании $(k, \pi_{k+1}^*(t - p_k))$, а второе – при расписании $(\pi_{k+1}^*(t), k)$.

Отметим, что при $k = n$ для любой точки t выполняется $\pi_n^*(t) = \{n\}$ и $F_n^*(t) = \max\{0, p_n - t + t_0\}$.

Расписание $\pi_1^*(d_n)$ является оптимальным для исходного примера I. Поэтому для применения данного подхода необходимо выполнение условия $d_n \in \mathbb{Z}$. Если $d_n \notin \mathbb{Z}$, тогда построим и решим пример $I' = \langle \{p_j, d'_j\}_{j \in N}, t'_0 \rangle$, где $d'_j = d_j - \Delta$, $j \in N$, $t'_0 = t_0 - \Delta$ и $\Delta = d_n - \lfloor d_n \rfloor$. В этом случае, пример I' равен исходному примеру I и выполняется $d'_n \in \mathbb{Z}$.

Таким образом мы можем решать примеры, когда $p_j, d_j \in \mathbb{R}$, для всех $j = 1, 2, \ldots, n$.

Если $t \leq t_0$, тогда для параметрического примера $I_k(t)$ для любого $k \in N$ выполняется

$$d_j(t) \leq d_n(t) = t - t_0 \leq 0 < p_j, \ j \in N_k.$$

Таким образом, все требования множества N_k запаздывают при любом расписании, и SPT расписание $(n, n-1, \ldots, k)$ является оптимальным в этом случае.

Если $t \geq t_0 + 2\sum_{j=1}^{n} p_j$, тогда по предположению $d_j \in [t_0, t_0 + \sum_{j \in N} p_j]$ (раздел 5.1) выполняется $t + \sum_{j=1}^{n} p_j \geq t_0 + 2\sum_{j=1}^{n} p_j + d_n - d_1$ Отсюда следует

$$\sum_{j=k}^{n} p_j \leq \sum_{j=k}^{n} p_j \leq d_1 - d_n + t - t_0 = d_1(t) \leq d_j(t),\ j \in N.$$

Таким образом, при любом расписании требования множества N_{k+1} не запаздывают, и любое расписание является оптимальным со значением целевой функции равным 0.

6.3 Алгоритмы решения задачи $1 \mid\mid \sum T_j$

6.3.1 Алгоритм B-1

В данном разделе предлагается алгоритм решения примеров в случае, когда параметры требований удовлетворяют условиям:

$$\begin{cases} p_1 \geq p_2 \geq \ldots \geq p_n; \\ d_1 \leq d_2 \leq \ldots \leq d_n; \\ d_n - d_1 \leq p_n. \end{cases} \tag{6.2}$$

То есть $d_1 \leq d_2 \leq \ldots \leq d_n \leq d_1 + p_n$. Рассматриваемый случай является подслучаем случая (6.1) при $\Bbbk = 1$ (т.е. $N = \mathcal{M}_1$).

Согласно лемме 6.1 в случае (6.2) существует оптимальное расписание π^*, при котором для любого требования $k \in N$ выполняется:

(a) либо $(k \to j)_{\pi^*}$ для всех $j \in \{k+1, \ldots, n\}$;

(b) либо $(j \to k)_{\pi^*}$ для всех $j \in \{k+1, \ldots, n\}$.

Данное свойство оптимальных расписаний в рассматриваемом случае позволяет использовать алгоритм B-1, с помощью которого может быть найдено оптимальное расписание за $O(n \sum p_j)$ операций. Для упрощения записи при описании алгоритма будем предполагать, что шаги 1 и 3–8 алгоритма выполняются для каждой целочисленной точки $t \in [t_0, d_n]$.

Алгоритм 6.2 B-1

1: $\pi_n(t) := (n)$, $F_n^*(t) := \max\{0, p_n - t + t_0\}$;
2: **for all** $k = n-1, n-2, \ldots, 1$ **do**
3: $\quad \pi^1 := (k, \pi_{k+1}^*(t - p_k))$;
4: $\quad \pi^2 := (\pi_{k+1}^*(t), k)$;
5: $\quad F(\pi^1) := \max\{0, p_k - d_k(t)\} + F_{k+1}^*(t - p_k)$;
6: $\quad F(\pi^2) := F_{k+1}^*(t) + \max\{0, \sum_{j=k}^{n} p_j - d_k(t)\}$;
7: $\quad F_k^*(t) := \min\{F(\pi^1), F(\pi^2)\}$;
8: $\quad \pi_k^*(t) := \arg\min\{F(\pi^1), F(\pi^2)\}$;
9: **end for**
10: RETURN расписание $\pi_1^*(d_n)$ и значение целевой функции $F_1^*(d_n)$.

Теорема 6.1 *Алгоритм B-1 находит оптимальное расписание для случая (6.2) за $O(n \sum p_j)$ операций.*

Доказательство. Оптимальность расписания $\pi_1^*(d_n)$ для исходного примера I следует из леммы 6.1. Действительно, так как оптимальное расписание для примеров $I_n(t)$, состоящих из одного требования, строится тривиальным образом (шаг 1) и для примеров $I_k(t)$ ($k \in N$ и $t \in [t_0, d_n]$) существует оптимальное расписание, при котором требование k обслуживается либо до всех, либо после всех требований множества $\{k+1, \ldots, n\}$, то расписание $\pi_k^*(t)$ является оптимальным для примера $I_k(t)$ ($k \in N$ и $t \in [t_0, d_n]$). Так как пример $I_1(d_n)$ равен исходному примеру I, то $\pi_1^*(d_n)$ является оптимальным расписанием для примера I.

Для фиксированных значений $k \in N$ и $t \in [t_0, d_n]$ шаг алгоритма выполняется за $O(1)$ операций. Согласно предположению из раздела 5.1 без ограничения общности рассматриваются только те примеры, для которых выполняется

$$d_j \in [t_0, t_0 + \sum_{j=1}^{n} p_j], \quad j \in N.$$

Таким образом, для каждого фиксированного k, $k = n, n-1, \ldots, 1$, необходимо перебрать не более $\sum_{j=1}^{n} p_j$ целочисленных точек t. Следовательно, трудоёмкость алгоритма B-1 составляет $O(n \sum p_j)$ операций. □

В заключение данного раздела представлен полиномиально разрешимый подслучай случая (6.2). В разделе 5.3 был рассмотрен алгоритм FA (алгоритм 5.5) трудоёмкости $O(n^3)$ операций построения оптимального расписания для случая, когда при любом расписании π выполнены следующие условия:

(i) $|D(\pi)| = m$, где m – некоторая константа $(0 \le m \le n)$;

(ii) требования множества $D(\pi)$ упорядочены в конце расписания π,

где $D(\pi)$ – множество запаздывающих при расписании π требований.

Рассмотрим пример I, параметры требований которого удовлетворяют условиям 6.2. Условие (ii) для любого расписания примера I выполнены, поскольку $d_n - d_1 \le p_n = \min\limits_{j \in N} p_j$. Далее, рассмотрим совокупность параметрических примеров $I(t) = \langle \{p_j, d_j(t)\}_{j \in N}, 0 \rangle, \quad t \in \mathbb{R}$, где, напомним, $d_j(t) = d_j - d_n + t - t_0$ для любого $j \in N$.

Для k, $k = 1, 2, \ldots, n$, определим две точки на временно́й оси τ_k и Θ_k. Для всех $t \le \tau_k$ при любом расписании π для параметрического примера $I(t)$ запаздывает более k требований. То есть для всех $t \le \tau_k$ для любого $\pi \in \Pi(I(t))$ выполняется $|D(\pi)| > k$. В ситуации $k = n$ будем полагать $\tau_n = -\infty$.

Для всех $t \ge \Theta_k$ при любом расписании π для параметрического примера $I(t)$ запаздывает менее k требований. То есть для всех $t \ge \Theta_k$ для любого $\pi \in \Pi(I(t))$ выполняется $|D(\pi)| < k$. В ситуации $k = 0$ будем полагать $\Theta_0 = +\infty$.

Найдём аналитические выражения для точек τ_k и Θ_k для примеров в случае (6.2). В граничном случае $k = 0$ величина τ_0 вычисляется следующим образом: $\tau_0 = \sum\limits_{i=1}^{n} p_i + t_0$. В случае $k = 1$ величина Θ_1 вычисляется следующим образом: $\Theta_1 = \sum\limits_{i=1}^{n} p_i + d_n - d_1 + t_0$. Данные точки находятся из условий $d_n(t) = \sum\limits_{i=1}^{n} p_i$ и $d_1(t) = \sum\limits_{i=1}^{n} p_i$, соответственно.

Для вычисления точек τ_k, $1 \le k \le n-1$, рассмотрим расписание π вида

$$\pi = (k+1, k+2, \ldots, n, k, k-1 \ldots, 1).$$

Найдём точку t' из условия $d_n(t') = \sum\limits_{i=k+1}^{n} p_i$, то есть $t' = \sum\limits_{i=k+1}^{n} p_i + t_0$. При данном расписании в точке $t' + \epsilon$ (ϵ – бесконечно малая величина) для примера $I(t')$ запаздывают требования множества $\{1, 2, \ldots, k\}$, суммарная продолжительность которых наибольшая среди любых других k требований. Поскольку последним незапаздывающим требованием является требование n, директивный срок которого максимален, то при любом другом расписании для точек $t \le t'$ будет запаздывать более k требований. Поэтому, $\tau_k = \sum\limits_{i=k+1}^{n} p_i + t_0$.

Для вычисления точек Θ_k, $2 \le k \le n$, рассмотрим расписание π вида

$$\pi = (n-k+1, n-k, \ldots, 1, n, n-1, \ldots, n-k+2).$$

Найдём точку t' из условия $d_1(t') = \sum\limits_{i=1}^{n-k+1} p_i$, то есть $t' = \sum\limits_{i=1}^{n-k+1} p_i + d_n - d_1 + t_0$. Правее требования 1 при данном расписании обслуживаются требования множества $\{n-k+2, n-k+3 \ldots, n\}$, суммарная продолжительность которых наименьшая среди любых других $k-1$ требований. Поскольку директивный срок требования 1 минимален, то при любом другом расписании для точек $t \ge t'$ будет запаздывать менее k требований. Следовательно, $\Theta_k = \sum\limits_{i=1}^{n-k+1} p_i + d_n - d_1 + t_0$.

Рассмотрим интервалы $(\tau_k, \Theta_k]$, $0 \le k \le n$, которые задают некоторое покрытие множества $\mathbb{R}$. Если существует такая точка t, что она покрывается одним и только одним интервалом, скажем с номером m, то можно утверждать, что при любом расписании для примера $I(t)$ запаздывает только m требований. Следовательно, алгоритм FA построит оптимальное расписание для такого примера за полиномиальное время $O(n^3)$ операций.

Для иллюстрации, рассмотрим следующий пример I с $n = 5$, $t_0 = 0$ и со следующими параметрами требований:

j	1	2	3	4	5
p_j	15	14	13	12	11
d_j	30	31	32	33	34

Для данного примера имеем

k	0	1	2	3	4	5
τ_k	65	50	36	23	11	$-\infty$
Θ_k	$+\infty$	69	58	46	33	19

Мы видим, что точки множеств $(-\infty, 11]$, $(19, 23]$, $(33, 36]$, $(46, 50]$, $(58, 65]$ и $(69, +\infty)$ покрываются только одним из полученных интервалов. Следовательно, оптимальное расписание для примеров $I(t)$ для точек t из указанных множеств может быть построено за полиномиальное время. Возьмем, например, точку $t = 34$. Пример $I(34)$ равен исходному примеру I, так как $d_n = 34$. При любом расписании для примера I запаздывает ровно три требования. Это означает, что пример I полиномиально разрешим. Взяв, например, точку $t = 22$ получим пример с директивными сроками $17, 18, 19, 20, 21, 22$, который также разрешим за полиномиальное время. Для данного примера при любом расписании запаздывает ровно два требования.

6.3.2 Алгоритм B-$\mathbb{k}$

В данном разделе представлен алгоритм решения примеров в случае (6.1). Пусть $\mathcal{M}_1, \ldots, \mathcal{M}_{\mathbb{k}}$ будет разбиением множества требований N исходного примера I в случае (6.1), полученного с помощью *процедуры разбиения* (алгоритм 6.1), $\mathcal{M}_\nu = \{\alpha_\nu, \alpha_\nu + 1, \ldots, \beta_\nu\}$, $\nu = 1, \ldots, \mathbb{k}$. Тогда условия (6.1) можно переписать в следующем виде:

$$\begin{cases} d_1 \le d_2 \le \ldots \le d_n; \\ p_1 \ge p_2 \ge \ldots \ge p_n; \\ d_{\beta_1} - d_{\alpha_1} \le p_{\beta_1}, & \alpha_1 = 1; \\ d_{\beta_2} - d_{\alpha_2} \le p_{\beta_2}, & \alpha_2 = \beta_1 + 1; \\ \ldots \\ d_{\beta_{\mathbb{k}}} - d_{\alpha_{\mathbb{k}}} \le p_{\beta_{\mathbb{k}}}, & \beta_{\mathbb{k}} = n. \end{cases} \tag{6.3}$$

Согласно леммам 6.1 и 6.2, при построении оптимального расписания можно исключить из рассмотрения такие расписания $\pi \in \Pi(I)$, при которых выполняется одно из следующих условий:

(a) существует такое требование $k \in N$, что $(i \to k \to j)_\pi$ для некоторых $i, j \in \mathcal{M}_\nu$, где $\gamma(k) < \nu$ и $k < \min\{i, j\}$;

(b) существует такое требование $k \in N$, что $(j \to k \to (k+1))_\pi$ для некоторого $j \in \{k+1, \ldots, n\}$.

Данное свойство оптимальных расписаний в рассматриваемом случае позволяет построить алгоритм B-$\mathbb{k}$, который находит оптимальное расписание за $O(\mathbb{k}n \sum p_j)$ операций. Алгоритм B-$\mathbb{k}$ является обобщением алгоритма B-1 на случай $\mathbb{k} > 1$. При построении оптимального расписания на каждом шаге (для каждого $k \in N$ и $t \in [t_0, d_n]$) для требования k необходимо проверять более двух позиций при оптимальном расписании. Отметим, что количество проверяемых позиций меньше или равно $\mathbb{k} + 1$.

Для описания алгоритма B-$\mathbb{k}$ нам понадобятся дополнительные обозначения для представления структуры расписаний $\pi_k^*(t)$. Пусть $G_k(t)$ будет упорядоченным множеством наборов из четырёх элементов $\langle \pi_i, \nu_i, f_i, P_i \rangle$, $i = 1, \ldots, g$, $g := |G_k(t)| \le \mathbb{k}$, где

(1) π_i, $i = 1, \ldots, g$, являются подрасписаниями расписания $\pi_k^*(t)$ такие, что

a) $\pi_k^*(t) = (\pi_1, \pi_2, \ldots, \pi_g)$;

b) $k \in \{\pi_1\}$;

c) каждое подмножество требований $\mathcal{M}_\nu \bigcap \{k, \ldots, n\}$, $\nu = \gamma(k), \ldots \mathbb{k}$, содержится не более чем в одном подрасписании;

d) никакое подрасписание не может быть далее разбито на два подрасписания так, чтобы условия a), b) и c) были выполнены.

(2) $\nu_i = \min\limits_{j \in \{\pi_i\}} \{\gamma(j)\}$;

(3) P_i – общее время обслуживания требований подрасписания π_i, т.е. $P_i = \sum\limits_{j \in \{\pi_i\}} p_j$;

(4) f_i – суммарное запаздывание требований подрасписания π_i.

Опишем процесс построения оптимального расписания $\pi_k^*(t)$ для примера $I_k(t)$ ($k \in N$, $t \in [t_0, d_n]$). Пусть примеры $I_j(t)$, $j = k+1, \ldots, n$, уже решены, т.е. известны расписания $\pi_j^*(t)$ и соответствующие множества $G_j(t)$ для целочисленных значений $t \in [t_0, d_n]$. Для построения расписания $\pi_k^*(t)$ нам необходимо знать возможные позиции требования k в данном расписании такие, что требование k не обслуживается между двумя требованиями i и j ($k < \min\{i, j\}$) из одного и того же подмножества $\mathcal{M}_\nu$ ($\nu \geq \gamma(k)$) и требование k не обслуживается между некоторым требованием j ($k < j$) и требованием $(k+1)$. Искомые позиции для требования k могут быть найдены при анализе расписания $\pi_{k+1}(t)$ за $O(n)$ операций. Для этого мы будем использовать информацию, содержащуюся в множествах $G_j(t)$ ($j > k$, $t \in [t_0, d_n]$) и собранную на предыдущих шагах алгоритма (при этом операция поиска возможных позиций будет требовать $O(\mathbb{k})$ операций).

Через $\sum\limits_q(X)$ будем обозначать операцию суммирования $\sum\limits_{i=1}^{q} X_i$ для некоторых пронумерованных величин X. Вместо символа X мы будем подставлять символы P и f.

Согласно леммам 6.1 и 6.2 позиции для требования k между двумя требованиями r, q могут быть исключены из рассмотрения, если $r, q \in \bar{\pi}_i$ для некоторого $1 \leq i \leq g$. Следовательно, имеется только $g+1$ позиция для требования k: до всех требований из множества $\{k+1 \ldots, n\}$, между требованиями каждой пары подрасписаний π_{i-1} и π_i и после всех требований множества $\{k+1, \ldots, n\}$. Если позиция между π_{i-1} и π_i является оптимальной, то присваиваем

$$\pi_k^*(t) := \left(\pi_1, \ldots, \pi_{i-1}, k, \pi^*_{\alpha_{\nu_i}}\left(t - \sum\nolimits_{i-1}(P) - p_k\right)\right).$$

Расписание $\pi^*_{\alpha_{\nu_i}}(t - \sum_{i-1}(P) - p_k)$ является оптимальным для множества требований $\{\pi_i\} \bigcup \{\pi_{i+1}\} \bigcup \ldots \bigcup \{\pi_g\}$ потому, что это множество равно множеству $\mathcal{M}_{\alpha_{\nu_i}} \bigcup \ldots \bigcup \mathcal{M}_{\alpha_{\Bbbk}}$, и оптимальное расписание для этого множества построено на предыдущих шагах алгоритма.

При этом, множество $G_k(t)$ строится следующим образом. Если требование k ставится на обслуживание между требованиями подрасписаний π_{i-1} и π_i, то все требования с меньшими номерами могут быть обслужены перед требованием k, только если они обслуживаются перед всеми требованиями множества $\{k+1, \ldots, n\}$. Это следует из леммы 6.2. Следовательно, мы можем объединить требования подрасписаний $\pi_1, \ldots, \pi_{i-1}$ в одно новое подрасписание в множестве $G_k(t)$. Таким образом,

$$G_k(t) := \{\big\langle (\pi_1, \ldots, \pi_{i-1}, k), \gamma(k), \textstyle\sum_{i-1}(P) + p_k, \\ \textstyle\sum_{i-1}(f) + \max\{0, \sum_{i-1}(P) + p_k - d_k(t)\}\big\rangle\} \bigcup G_{\alpha_{\nu_i}}(t - \textstyle\sum_{i-1}(P) - p_k).$$

Приведём формальное описание алгоритма B-$\Bbbk$. Для упрощения записи при описании алгоритма будем предполагать, что строки 1 и 4–9 алгоритма выполняются для каждой целочисленной точки $t \in [t_0, d_n]$.

Теорема 6.2 *Алгоритм B-$\Bbbk$ строит расписание за $O(\Bbbk n \sum p_j)$ операций. В случае, когда исходные данные удовлетворяют (6.3), то построенное расписание будет оптимальным.*

Алгоритм 6.3 B-$\Bbbk$

1: $\pi^*_n(t) := (n)$, $F^*_n(t) := \max\{0, p_n + t_0 - t\}$, $G_n(t) = \{\langle \pi_n(t), \Bbbk, F_n(t), p_n \rangle\}$;
2: **for all** $\nu = \Bbbk, \Bbbk - 1, \ldots, 1$ **do**
3: **for all** $k = \beta_\nu, \beta_\nu - 1, \ldots, \alpha_\nu$, $k < n$ **do**
4: **for all** $i = 1, 2, \ldots, g+1$ **do**
5: $\pi^i := (\pi_1 \ldots, \pi_{i-1}, k, \pi^*_{\alpha_{\nu_i}}(t - \sum\limits_{i-1}(P) - p_k))$;
6: $F(\pi^i) := \sum\limits_{i-1}(f) + \max\{0, \sum\limits_{i-1}(P) + p_k - d_k(t)\} + F^*_{\alpha_{\nu_i}}(t - \sum\limits_{i-1}(P) - p_k)$;
7: **end for**
8: $i^* := \arg \min\limits_{i=1,\ldots,g+1}\{F(\pi^i)\}$; $\pi^*_k(t) := \pi^{i^*}$; $F^*_k(t) := F(\pi^{i^*})$;
9: $G_k(t) := \Big\{\Big\langle (\pi_1, \ldots, \pi_{i^*-1}, k), m, \sum\limits_{i^*-1}(P) + p_k, \sum\limits_{i^*-1}(f) + \max\{0, \sum\limits_{i^*-1}(P) + p_k - d_k(t)\}\Big\rangle\Big\} \bigcup G_{\alpha_{\nu_{i^*}}}(t - \sum\limits_{i^*-1}(P) - p_k)$;
10: **end for**
11: **end for**
12: RETURN расписание $\pi^*_1(d_n)$ и значение целевой функции $F^*_1(d_n)$.

Доказательство. Каждый шаг алгоритма B-$\Bbbk$ организован в соответствии с леммами 6.1 и 6.2. Возможные позиции для требования k просматриваются при переборе элементов множества $G_{k+1}(t)$. Все остальные позиции могут быть исключены из рассмотрения согласно вышеуказанным свойствам. Таким образом, расписание $\pi_1^*(d_n)$ является оптимальным для случая (6.3).

Заметим, что каждый шаг алгоритма для фиксированных k, где $k = n, n-1, \ldots, 1$, и t выполняется за $O(\Bbbk)$ операций. Следовательно, алгоритм B-$\Bbbk$ выполняется за $O(\Bbbk n \sum p_j)$ операций. □

6.3.3 Алгоритм C-1

Пусть параметры требований удовлетворяют условиям:

$$\begin{cases} d_1 \le d_2 \le \ldots \le d_n; \\ d_n - d_1 \le 1; \\ t_0 \in \mathbb{Z}. \end{cases} \tag{6.4}$$

В данном разделе доказываются три леммы, на основе которых строится алгоритм решения данного случая с трудоёмкостью $O(n^2)$ операций. Будем пользоваться следующими обозначениями: $S = t_0 + \sum\limits_{j \in N'} p_j$, $N' \subset N$, и $z = \lfloor d_n \rfloor$.

Лемма 6.3 *Пусть дан пример $I' = \{N', t_0\}$, где $N' \subseteq N$ и $S - p_j > z$ для всех $j \in N'$. В случае (6.4) существует оптимальное расписание $\pi^* \in \Pi^*(I')$ такое, что требование $k = \arg\max\limits_{j \in N'}\{d_j : p_j = \max\limits_{i \in N'} p_i\}$ обслуживается на последней позиции при расписании π^*, т.е. выполняется $(j \to k)_{\pi^*}$ для всех $j \in N' \setminus \{k\}$.*

Доказательство. Предположим, что существует оптимальное расписание π^*, при котором требование k обслуживается не на последней позиции, т.е. $\pi^* = (\pi_1, k, \pi_2, i, j)$, где $j \ne k$ и множество $\{\pi_2, i\}$ может быть пустым.

Построим расписание $\pi' = (\pi_1, j, \pi_2, i, k)$. Покажем, что при расписании π' выполняется либо $F(\pi') = F(\pi^*)$, что означает оптимальность расписания π', либо $F(\pi') < F(\pi^*)$, что противоречит оптимальности расписания π^*. Это противоречие означает, что при любом оптимальном расписании требование k обслуживается на последней позиции.

Заметим, что $p_k + d_k = \max\limits_{q \in N'}\{p_q + d_q\}$. Это следует из того, что $p_q \in \mathbb{Z}^+$ и $\mid d_\alpha - d_\beta \mid \le 1$ для любых требований $\alpha, \beta \in N'$.

Если $p_k = p_j$, тогда $d_k \ge d_j$ и

(a) $F(\pi') - F(\pi^*) = 0$, если $C_k(\pi) \geq d_j$;

(b) $F(\pi') - F(\pi^*) \leq d_j - d_k \leq 0$, если $C_k(\pi) < d_j$.

Если $p_k > p_j$, то рассмотрим два следующих случая. Заметим, что поскольку продолжительности обслуживания требований целочислены, то $p_k \geq p_j + 1$.

a) Пусть $\{\pi_2, i\} = \emptyset$. В этом случае $\pi^* = (\pi_1, k, j)$ и $\pi' = (\pi_1, j, k)$, т.е. требование k обслуживается при расписании π^* непосредственно перед требованием j. Так как $S - p_q > z$ для всех $q \in N'$, то выполняется

$$C_k(\pi^*) = S - p_j > z + 1 \quad \text{и} \quad C_j(\pi') = S - p_k \geq z + 1.$$

Таким образом, учитывая $d_q \leq z + 1$ для всех $q \in N'$, получаем, что требования k и j запаздывают при обоих расписаниях π^* и π', и, следовательно, выполняется $F(\pi') - F(\pi^*) = p_j - p_k < 0$.

b) Пусть $\{\pi_2, i\} \neq \emptyset$, т.е. при расписании π^* существует такое требование i, что $(k \to i \to j)_{\pi^*}$. В этом случае выполняется:

$$\begin{array}{ll} T_k(\pi^*) = \max\{0, C_k(\pi^*) - d_k\}, & T_k(\pi') = S - p_k, \\ T_i(\pi^*) = S - p_j - d_i, & T_i(\pi') = S - p_k - d_i \\ T_j(\pi^*) = S - d_j, & T_j(\pi') = \max\{0, C_k(\pi^*) - p_k + p_j - d_j\}, \end{array}$$

а также

$$F(\pi') - F(\pi^*) \leq T_k(\pi') + T_i(\pi') + T_j(\pi') - (T_k(\pi^*) + T_i(\pi^*) + T_j(\pi^*)).$$

Рассмотрим следующие подслучаи:

b.1) Пусть $C_k(\pi^*) \leq d_k$. В этом случае выполняется $C_j(\pi') \leq d_j$ и

$$F(\pi') - F(\pi^*) \leq (p_j + d_j) - (p_k + d_k) < 0.$$

b.2) Пусть $C_k(\pi^*) > d_k$ и $C_j(\pi') \leq d_j$. В этом случае выполняется $F(\pi') - F(\pi^*) \leq (d_j - C_k(\pi^*)) - (p_k - p_j)$. Так как $C_k(\pi) > d_k$, то выполняется $d_j - C_j(\pi^*) < d_j - d_k \leq 1$. Так как $p_k - p_j \geq 1$, то $F(\pi') - F(\pi^*) \leq 0$.

b.3) Пусть $C_k(\pi^*) > d_k$ и $C_j(\pi') > d_j$. В этом случае выполняется

$$F(\pi') - F(\pi^*) \leq 2(p_j - p_k) < 0.$$

Таким образом, $F(\pi') \leq F(\pi^*)$. Требование k обслуживается на последней позиции при оптимальном расписании π'. □

Лемма 6.4 *Пусть дан пример* $I' = \{N', t_0\}$ *такой, что* $N' \subseteq N$ *и множество* $\bar{N} = \{i \in N' : S - p_i \leq z\}$ *не пусто. В случае (6.4) существует оптимальное расписание* $\pi^* \in \Pi^*(I')$*, при котором на последней позиции обслуживается требование* k *такое, что* $S - p_k \leq z + 1$.

Доказательство. Предположим, что существует оптимальное расписание π^*, при котором на последнем месте обслуживается требование j такое, что $S - p_j > z + 1$. То есть $\pi^* = (\pi_1, q, \pi_2, i, j)$, где q есть некоторое требование из множества $\bar{N}$, и множество $\{\pi_2, i\}$ может быть пустым. Построим расписание $\pi' = (\pi_1, j, \pi_2, i, q)$.

Покажем, что $F(\pi') \leq F(\pi^*)$, что означает оптимальность расписания π'. При этом, для требования q выполняется условие $S - p_q \leq z + 1$.

Поскольку $j \notin \bar{N}$ и $S - p_q \leq z$, то выполняется неравенство $p_q > p_j + 1$. Так как продолжительности обслуживания требований целочислены, то $p_q \geq p_j + 2$.

Рассмотрим следующие случаи:

a) пусть $\{\pi_2, i\} = \emptyset$. В этом случае $\pi^* = (\pi_1, q, j)$ и $\pi' = (\pi_1, j, q)$. То есть требование q обслуживается при расписании π^* непосредственно перед требованием k. Тогда

$$T_q(\pi^*) = S - p_j - d_q, \qquad T_q(\pi') = S - d_q,$$
$$T_j(\pi^*) = S - d_j, \qquad T_j(\pi') = \max\{0, S - p_q - d_j\}.$$

Так как $p_q \geq p_j + 2$, то выполняется

$$F(\pi') - F(\pi^*) = \max\{0, S - p_q - d_j\} - S + p_j + d_j < 0;$$

b) пусть $\{\pi_2, i\} \neq \emptyset$, т.е. при расписании π^* существует требование i такое, что $(q \to i \to j)_{\pi^*}$. В этом случае выполняется:

$$T_q(\pi^*) = \max\{C_q(\pi^*) - d_q, 0\}, \qquad T_q(\pi') = S - d_q,$$
$$T_j(\pi^*) = S - d_j, \qquad T_j(\pi') = 0,$$
$$T_i(\pi^*) = S - p_j - d_i, \qquad T_i(\pi') = \max\{S - p_q - d_i, 0\}.$$

Рассмотрим следующие подслучаи:

b.1) пусть $T_q(\pi^*) = 0$ и $T_i(\pi') = 0$. В этом случае

$$F(\pi') - F(\pi^*) \leq p_j + d_j - S + d_i - d_q.$$

Так как $d_i - d_q \leq 1$, $S - p_j > z + 1$ и $p_j \in \mathbb{Z}^+$, то выполняется

$$S - p_j \geq z + 2 \geq d_j + 1 \quad \text{и} \quad p_j + d_j - S \leq -1.$$

Следовательно $F(\pi') - F(\pi^*) \leq 0$;

b.2) пусть $T_q(\pi^*) > 0$ и $T_i(\pi') = 0$. В этом случае

$$F(\pi') - F(\pi^*) \le p_j + d_j - S + d_i - C_q(\pi^*).$$

Так как $C_q(\pi^*) > d_q$, то $d_i - C_q(\pi^*) \le d_i - d_q \le 1$.
Следовательно, $F(\pi') - F(\pi^*) \le 0$;

b.3) пусть $T_q(\pi^*) = 0$ и $T_i(\pi') > 0$. В этом случае

$$F(\pi') - F(\pi^*) \le (p_j + d_j) - (p_q + d_q).$$

Так как $p_j - p_q \le -2$ и $d_j - d_q \le 1$, то $F(\pi') - F(\pi^*) < 0$;

b.4) пусть $T_q(\pi^*) > 0$ и $T_i(\pi') > 0$. В этом случае

$$F(\pi') - F(\pi^*) \le (p_j + d_j) - (C_q(\pi^*) + d_q) \le (p_j + d_j) - (p_q + d_q) \le 0.$$

Таким образом, мы показали, что расписание π' также оптимально и на последней позиции при расписании π' обслуживается требование q такое, что $S - p_q \le z + 1$. □

Лемма 6.5 *Пусть дан пример $I' = \{N', t_0\}$, где $N' \subseteq N$ и $S \le z$. В случае (6.4) существует оптимальное расписание $\pi^* \in \Pi^*(I')$, при котором требование $k = \arg\max\limits_{j \in N'}\{d_j\}$ обслуживается на последней позиции.*

Доказательство. Поскольку $S \le z$ и $p_j \in \mathbb{Z}^+$, то при любом расписании $\pi \in \Pi(I')$ запаздывает не более одного требования. Следовательно, расписание, при котором на последней позиции обслуживается требование с максимальным директивным сроком среди требований множества N', является оптимальным. □

Приведём алгоритм решения задачи в случае (6.4). При описании алгоритма π_{edd} обозначает расписание, составленное из требований множества N', обслуживаемых в EDD порядке. Алгоритм C-1 состоит из четырёх основных шагов. Каждый из шагов алгоритма организован в соответствии с леммами 6.3 – 6.5. Первоначально полагаем $N' = N$ и $\pi^* = \varnothing$. На первом шаге (строки 2–5) среди требований множества N' выбирается требование с максимальной продолжительностью (если таких требований несколько, то выбирается требование с максимальным номером) и помещается перед уже упорядоченными при расписании π^* требованиями множества $N \setminus N'$. Шаг 1 (строки 2–5) выполняется до тех пор пока множество $\{j \in N' : S - p_j \le z\}$ пусто. На втором шаге (строки 6–8) проверяется тривиальная ситуация, когда $|N'| \le 1$. На третьем шаге (строки 9–13) строятся расписания π_{ij} для

всех $j \in N'$ таких, что $S - p_j \leq z + 1$, и для всех $i \in N' \setminus \{j\}$. На четвертом шаге (строка 14) строится оптимальное расписание, при котором сначала обслуживаются требования наилучшего из расписаний π_{ij}, а затем требования расписания, построенного на шаге 1.

Теорема 6.3 *Алгоритм C-1 строит расписание за $O(n^2)$ операций. В случае (6.4), когда исходные данные удовлетворяют (6.4), построенное расписание будет оптимальным.*

Доказательство. Оптимальность расписания π^*, построенного алгоритмом C-1 следует из того, что первый шаг алгоритма организован в соответствии с леммой 6.3 и на третьем шаге расписания π_{ij} строятся для всех возможных пар требований i, j. При этом, требование j выбирается в соответствии с леммой 6.4. Так как величины p_i и p_j целочислены и $S - p_j \leq z + 1$, то $S - p_i - p_j \leq z$, и требования множества $N' \setminus \{i, j\}$ могут быть обслужены в EDD порядке согласно лемме 6.5.

Алгоритм 6.4 C-1

1: $S := t_0 + \sum_{j=1}^{n} p_j$, $N' := N$, $\pi^* := \emptyset$, $\pi_{edd} := (1, 2, \ldots, n)$;
2: **while** $\{j \in N' : S - p_j \leq z\} = \emptyset$ **and** $N' \neq \emptyset$ **do**
3: $\quad \pi^* := (k, \pi^*)$, где $k := \arg\max_{j \in N'} \{d_j : p_j = \max_{i \in N'} p_i\}$;
4: $\quad S := S - p_k$, $N' := N' \setminus \{k\}$, $\pi_{edd} := \pi_{edd} \setminus \{k\}$;
5: **end while**
6: **if** $N' = \{j\}$ **then**
7: $\quad \pi^* := (j, \pi^*)$, **stop;**
8: **end if**
9: **for all** $j \in N'$ таких, что $S - p_j \leq z + 1$ **do**
10: $\quad$ **for all** $i \in N' \setminus \{j\}$ **do**
11: $\quad\quad \pi_{ij} := (\pi_{edd} \setminus \{i, j\}, i, j)$;
12: $\quad$ **end for**
13: **end for**
14: $\pi^* := (\pi, \pi^*)$, где $\pi := \arg\min_{i,j} F(\pi_{ij})$.

Вычислим трудоёмкость алгоритма. Шаг 1 алгоритма выполняется за $O(n \log n)$ операций. На шаге 3 на основе расписания π_{edd} строится не более, чем n^2 расписаний, из которых на шаге 4 выбирается наилучшее. Поскольку в каждом расписании π_{ij} запаздывает не более, чем три последних по порядку требования, то для вычисления величины $F(\pi_{ij})$ требуется $O(1)$ операций. Следовательно шаг 4 выполняется за $O(n^2)$ операций. Таким образом, трудоёмкость алгоритма C-1 составляет $O(n^2)$ операций. □

С учётом замечания, высказанного в разделе 5.1 относительно пространственного представления множества примеров, можно утверждать, что алгоритм C-1 находит оптимальное расписание в случае, если в условиях (6.4) второе неравенство заменить на $d_n - d_1 \leq \mathbf{НОД}\,(p_1, \ldots, p_n)$.

6.3.4 Алгоритм B-n

Рассмотрим другой "предельный случай" исследуемой задачи. Пусть параметры требований удовлетворяют условиям

$$d_j - d_{j-1} > p_j,\ j = 2, 3, \ldots, n. \tag{6.5}$$

Поскольку $p_j > 0$, то выполняется $d_1 < d_2 < \ldots < d_n$. Отметим, что примеры в случае (6.1) при $\Bbbk = n$ удовлетворяют условиям (6.5).

Покажем, что в случае (6.5) существует оптимальное расписание π^*, при котором требование j^* обслуживается на первой позиции из множества L (см. раздел 5.2). На основе этого свойства приведём алгоритм решения примеров в случае (6.5) с трудоёмкостью $O(n^2)$ операций.

Лемма 6.6 *Для примеров I в случае* (6.5) *существует оптимальное расписание $\pi^* \in \Pi^*(I)$, при котором требование j^* обслуживается на первой позиции из множества $L(I)$.*

Доказательство. Пусть $L(I) = \{l_1, l_2, \ldots, l_m\}$, где $l_1 < l_2 < \ldots < l_m$. Рассмотрим две соседние позиции l_i и l_{i+1} из множества $L(I)$, $i = 1, \ldots, m-1$. Далее в доказательстве будем использовать запись $\alpha = l_i$ и $\beta = l_{i+1}$, а также, $S_\alpha = t_0 + \sum\limits_{j=1}^{\alpha} p_j$ и $S_\beta = t_0 + \sum\limits_{j=1}^{\beta} p_j$. Согласно определению множества $L(I)$, выполняется $j^* \leq \alpha < \beta$. Согласно определению требования j^* выполняется $p_{j^*} > p_j$ для всех $j \in \{j^* + 1, \ldots, n\}$.

Пусть $\pi_\alpha = (\pi_1, j^*, \pi_2)$ и $\pi_\beta = (\pi_1', j^*, \pi_2')$ будут расписаниями с наименьшим значением суммарного запаздывания, при которых требование j^* обслуживается на позициях α и β, соответственно, т.е.

- $\pi_1 \in \Pi^*(\{1, \ldots, \alpha\} \setminus \{j^*\}, t_0)$, $\pi_2 \in \Pi^*(\{\alpha + 1, \ldots, n\}, S_\alpha)$,
- $\pi_1' \in \Pi^*(\{1, \ldots, \beta\} \setminus \{j^*\}, t_0)$, $\pi_2' \in \Pi^*(\{\beta + 1, \ldots, n\}, S_\beta)$.

При этом, согласно теореме 5.1, для позиций α и β выполняется

- $S_\alpha \leq d_{\alpha+1}$ и $d_j + p_j \leq S_\alpha$ для всех $j \in \{j^* + 1, \ldots, \alpha\}$;

- $S_\beta \le d_{\beta+1}$ и $d_j + p_j \le S_\beta$ для всех $j \in \{j^*+1, \ldots, \beta\}$,

доопределив $d_{n+1} := +\infty$.

Рассмотрим случай $d_{j^*} \ge S_\alpha$. С одной стороны, выполняется

$$S_\alpha \le d_{j^*} < d_{\alpha+1} \quad \text{и} \quad d_\beta + p_\beta \le S_\beta.$$

Таким образом, $d_\beta - d_{j^*} < S_\beta - S_\alpha$. С другой стороны, согласно условиям (6.5), выполняется

$$d_\beta - d_{j^*} \ge p_{\alpha+1} + p_{\alpha+2} + \ldots + p_\beta = S_\beta - S_\alpha.$$

Это противоречие означает, что $|L(I)| = 1$, т.е. множество L содержит только один элемент, и лемма в случае $d_{j^*} \ge S_\alpha$ верна.

Рассмотрим случай $d_{j^*} < S_\alpha$. Покажем, что расписание π_1' имеет вид $\pi_1' = (\pi_1, \alpha+1, \alpha+2, \ldots, \beta)$. Согласно выбору α и j^*, выполняется

$$C_{\alpha+1}(\pi_1') = S_\alpha - p_{j^*} + p_{\alpha+1} < S_\alpha \le d_{\alpha+1}.$$

Следовательно требование $\alpha+1$ не запаздывает при расписании π_1', то есть $T_{\alpha+1}(\pi_1') = 0$. Из условий (6.5) следует $T_j(\pi_1') = 0$ для всех требований $j \in \{\alpha+1, \ldots, \beta\}$. Таким образом, требования множества $\{\alpha+1, \ldots, \beta\}$ не запаздывают при расписании π_1' и могут быть обслужены при расписании π_1' в указанном порядке. Построим расписание $\pi' = (\pi_1, j^*, \bar{\pi}', \pi_2')$, где

$$\bar{\pi}' = (\alpha+2, \alpha+3, \ldots, \beta, \alpha+1).$$

Поскольку расписание π_α является расписанием с наименьшим значением целевой функции среди расписаний, при которых требование j^* обслуживается на позиции α, то выполняется $F(\pi_\alpha) \le F(\pi')$. Заметим, что момент начала обслуживания частичного расписания $\bar{\pi}'$ при расписании π' равен S_α. Покажем, что справедливо неравенство $F(\pi') < F(\pi_\beta)$. Как было показано, для всех требований $j \in \{\alpha+1, \ldots, \beta\}$ выполняется $T_j(\pi_\beta) = 0$. Следовательно,

$$F(\pi') - F(\pi_\beta) = \Delta + F(\bar{\pi}', S_\alpha),$$

где $\Delta = T_{j^*}(\pi') - T_{j^*}(\pi_\beta)$.

Так как $d_{j^*} < S_\alpha$, то требование j^* запаздывает при обоих расписаниях π' и π_β. Таким образом, $\Delta = -\sum_{j=\alpha+1}^{\beta} p_j = S_\alpha - S_\beta$. Вычислим величину $F(\bar{\pi}', S_\alpha)$. Согласно условиям (6.5) и $S_\alpha < d_{\alpha+1}$, выполняется $T_j(\bar{\pi}', S_\alpha) = 0$ для всех $j \in \{\alpha+2, \ldots, \beta\}$. Поскольку $d_{\alpha+1} + p_{\alpha+1} \le S_\beta$, то выполняется

$T_{\alpha+1}(\bar{\pi}', S_\alpha) = S_\beta - d_{\alpha+1}$. Таким, образом, $F(\bar{\pi}', S_\alpha) = S_\beta - d_{\alpha+1}$. Следовательно,

$$F(\pi') - F(\pi_\beta) = S_\alpha - S_\beta + S_\beta - d_{\alpha+1} < 0.$$

Мы показали, что $F(\pi_\alpha) \leq F(\pi') < F(\pi_\beta)$. Так как α и β являются произвольными соседними позициями для требования j^*, то выполняется

$$F(\pi_{l_1}) < F(\pi_{l_2}) < \ldots < F(\pi_{l_m}),$$

где π_{l_i} является расписанием с наименьшим значением целевой функции среди всех расписаний, при которых требование j^* обслуживается на позиции l_i, $i = 1, 2, \ldots, m$.

Следовательно, в случае (6.5) существует оптимальное расписание π^*, при котором требование j^* обслуживается на первой позиции множества $L(I)$. □

Приведём алгоритм *B-n* построения оптимального расписания в случае (6.5). Данный алгоритм является модификацией алгоритма A на случай перебора только первой позиции из множества $L(I)$.

Алгоритм 6.5 Процедура **SequenceB-n**(I)

1: $S_l := t_0 + \sum_{j=1}^{l} p_j$, $l = 1, 2, \ldots, n$; $l^* := \arg\min\{l \in L(I)\}$;
2: $N' := \{1, \ldots, l^*\} \setminus \{j^*\}$, $t' := t_0$, $I' = \{N', t'\}$;
3: $N'' := \{l^* + 1, \ldots, n\}$, $t'' := S_{l^*}$, $I'' = \{N'', t''\}$;
4: $\pi^* :=$ (**SequenceB-n** (I'), j^*, **SequenceB-n** (I'')); RETURN π^*.

Алгоритм 6.6 B-n(I)

1: $\pi^* :=$ **SequenceB-n**(I).

Теорема 6.4 *Алгоритм B-n строит расписание за $O(n^2)$ операций. Когда параметры требований удовлетворяют (6.5), то построенное расписание является оптимальным*

Доказательство. Поскольку каждый подпример $\{N', t'\}$, $N' \subseteq N$, $t' \geq t_0$, удовлетворяет условиям (6.5), то оптимальность расписания π^*, построенного алгоритмом *B-n*, в случае (6.5) следует из теоремы 5.1 и леммы 6.6.

Каждый вызов процедуры **SequenceB-n** фиксирует позицию одного требования в расписании π^* и выполняется за $O(n)$ операций. Следовательно алгоритм *B-n* выполняется за $O(n^2)$ операций. Таким образом весь класс задач (6.1) нами рассмотрен и построены алгоритмы, трудоемкость которых не превышает $O(n^2 \sum p_j)$ операций. □

Глава 7

Исследование свойств и приложения алгоритма B-1

В данной главе рассматриваются свойства алгоритма B-1, изучение которых направлено на обнаружение новых разрешимых данным алгоритмом частных случаев задачи, а также на сокращение трудоёмкости алгоритма.

В разделе 7.1 рассматриваются расписания, обладающие *свойством* B-1, и доказывается, что если для исходного примера I существует оптимальное расписание, обладающее *свойством* B-1, то алгоритм B-1 находит оптимальное расписание для данного примера I. В этом разделе также приводится частный случай задачи, для которой с помощью доказанного свойства установлено, что алгоритм B-1 решает примеры данного случая.

В разделе 7.2 рассматриваются три NP-полные задачи разбиения: РАЗБИЕНИЕ, ЧЁТНО-НЕЧЁТНОЕ РАЗБИЕНИЕ и ЧЁТНО-НЕЧЁТНОЕ РАЗБИЕНИЕ С ОГРАНИЧЕНИЯМИ. В этом разделе приводятся схемы полиномиального сведения примеров задач о разбиении к *каноническим* примерам задачи $1 \,||\, \sum T_j$ и показывается, что для построенных *канонических* примеров существует оптимальное расписание, обладающее *свойством* B-1 (т.е. показывается, что алгоритм B-1 решает данные задачи о разбиении). Далее, учитывая некоторые особенности *канонических* примеров предлагается алгоритм B-1-*канонический*, который является модификацией алгоритма B-1. В разделе 7.3 рассматривается алгоритм B-1-*модифицированный*, который находит оптимальное расписание для тех же примеров, что и алгоритм B-1, путём построения кусочно-линейных функций $F_k^*(t)$, $k = 1, \ldots, n$, $t \in \mathbb{R}$.

7.1 Свойство B-1

Определение 7.1 *Расписание π назовем расписанием, обладающим свойством* B-1*, если для всех требований $k \in \{1, 2, \ldots, n-1\}$ при расписании π выполняется:*

(a) либо $(k \to j)_\pi$ для всех $j \in \{k+1, \ldots, n\}$;

(b) либо $(j \to k)_\pi$ для всех $j \in \{k+1, \ldots, n\}$.

Расписание π не обладает свойством B-1, если существует тройка требований i, j, k, таких, что $k < \min\{i, j\}$ и при расписании π выполняется $(i \to k \to j)_\pi$.

Множество всех расписаний, обладающих свойством B-1 для примера I, будем обозначать через $\Pi_{B-1}(I)$.

Пусть $\Pi^*_{B-1}(I) = \Pi_{B-1}(I) \bigcap \Pi^*(I)$. Таким образом, множество $\Pi^*_{B-1}(I)$ является множеством всех оптимальных расписаний, обладающих свойством B-1.

Для расписания π и требования $k \in \{\pi\}$ определим величины $\chi_k(\pi)$ следующим образом:

(a) $\chi_k(\pi) = 0$, если $k = n$ или существует такое требование $j > k$, что $(j \to k)_\pi$;

(b) $\chi_k(\pi) = 1$, в противном случае.

Для расписания $\pi \in \Pi_{B-1}(I)$ и требования $k \in N$ выполняется: $\chi_k(\pi) = 0$ в случае, если $k = n$ или требование k обслуживается при расписании π после всех требований множества $\{k+1, \ldots, n\}$; и $\chi_k(\pi) = 1$ в случае, если требование k обслуживается перед всеми требованиями множества $\{k+1, \ldots, n\}$.

Теорема 7.1 *Алгоритм* B-1 *находит оптимальное расписание для некоторого примера I задачи $1 \,||\, \sum T_j$ тогда и только тогда, когда множество $\Pi^*_{B-1}(I) \neq \emptyset$.*

Доказательство. Необходимое условие следует из схемы построения расписаний $\pi^*_k(t)$, $k \in N$, $t \in [t_0, d_n]$ алгоритмом B-1. Действительно, при построении расписания алгоритм B-1 на каждом шаге перебирает две возможные позиции для каждого требования $k \in N$: до всех требований множества

$\{k+1,\ldots,n\}$ и после этих требований. Таким образом, любое построенное алгоритмом B-1 расписание обладает свойством B-1, в том числе и оптимальное.

Докажем достаточное условие. Обозначим через π' расписание, построенное алгоритмом B-1. Покажем, что если $\Pi^*_{B-1}(I) \neq \emptyset$, то $\pi' \in \Pi^*_{B-1}(I)$.

Предположим противное. Пусть $\Pi^*_{B-1}(I) \neq \emptyset$, но $\pi' \notin \Pi^*_{B-1}(I)$, т.е. требования множества N упорядочены при расписании π' неоптимальным образом. Рассмотрим следующую схему построения множеств расписаний $\Pi_0, \Pi_1, \ldots, \Pi_n$. Пусть $\Pi_0 = \Pi^*_{B-1}(I)$ и

$$\Pi_j = \Pi_{j-1} \bigcap \{\pi : \pi \in \Pi^*(I) \text{ и } \chi_j(\pi) = \chi_j(\pi')\}, \quad j = 1,\ldots,n.$$

Если для некоторого требования $j \in N$ выполняется $\Pi_j \neq \emptyset$, то все требования множества $\{1,2,\ldots,j\}$ упорядочены при расписании π' таким же образом, что и при некотором оптимальном расписании, обладающим свойством B-1 (т.е. существует некоторое оптимальное расписание со свойством B-1, при котором каждое требование i из множества $\{1,2,\ldots,j\}$ либо предшествует, либо следует за всеми требованиями с бóльшими номерами так же, как и при расписании π'). Если $\Pi_j = \emptyset$ для некоторого $j \in N$, тогда для всех $i \in \{j+1,\ldots,n\}$ выполняется $\Pi_i = \emptyset$. Заметим, что при предположении $\pi' \notin \Pi^*_{B-1}(I)$ выполняется $\Pi_n = \emptyset$.

Пусть $j_0 = \min\{j \in N : \Pi_j = \emptyset\}$. Согласно нашему предположению, такое требование j_0 существует. Построим пример с множеством требований $N' = \{j_0+1,\ldots,n\}$ и моментом начала обслуживания $t'_0 = t_0 + \sum\limits_{j=1}^{j_0} \chi_j(\pi')p_j$. Из $\pi' \notin \Pi^*_{B-1}(I)$ следует, что требования множества N' обслужены при расписании π' с момента времени t'_0 неоптимальным образом. Заметим, что $N' \subset N$, т.е. $|N'| < |N|$.

Повторим вышеуказанные рассуждения для примера $\{N', t'_0\}$ и в силу нашего предположения $\pi' \notin \Pi^*_{B-1}(I)$ найдём подмножество требований N'' и момент начала обслуживания t''_0 такие, что требования множества N'' обслужены при расписании π' с момента времени t''_0 неоптимальным образом. При этом, $N'' \subset N'$ и так далее.

Продолжая подобные рассуждения, получим, что с некоторого момента времени требование n упорядочено неоптимальным образом. Это противоречит тому, что пример с множеством из одного требования с любого момента времени имеет единственное расписание обслуживания, которое и является оптимальным.

Из полученного противоречия следует, что $\pi' \in \Pi^*_{B-1}(I)$, т.е. алгоритмом B-1 построено оптимальное расписание. Теорема доказана. $\square$

Рассмотрим пример использования теоремы 7.1. Пусть параметры требований некоторого примера I удовлетворяют условиям:

$$\begin{cases} d_1 \le d_2 \le \ldots \le d_n; \\ d_n - d_1 \le \min\limits_{j \in N} p_j; \\ p_k \le p_j \text{ для всех } j > k, \text{ либо } p_k \ge p_j \text{ для всех } j > k, \quad k \in N. \end{cases} \tag{7.1}$$

Третье ограничение системы (7.1) означает, что продолжительность обслуживания любого требования $k \in N$ либо одновременно не больше, либо одновременно не меньше продолжительностей обслуживания всех требований $j \in \{k+1, \ldots, n\}$.

В случае (7.1), если для некоторого требования k выполняется $p_k \le p_j$ для всех $j > k$, то согласно условиям Эммонса существует оптимальное расписание π^*, при котором выполняется $(k \to j)_{\pi^*}$ для всех $j > k$. Если для некоторого требования k выполняется $p_k \ge p_j$ для всех $j > k$, то согласно лемме 6.1 существует оптимальное расписание π^*, при котором выполняется либо $(k \to j)_{\pi^*}$ для всех $j > k$, либо $(j \to k)_{\pi^*}$ для всех $j > k$.

Таким образом, в случае (7.1) существует оптимальное расписание, обладающее свойством B-1. Тогда, согласно теореме 7.1 алгоритм B-1 находит оптимальное расписание в случае (7.1), хотя пример может и не удовлетворять условиям (6.1).

7.2 Алгоритм решения задачи РАЗБИЕНИЕ

7.2.1 Постановка и полиномиальная сводимость задач разбиения

В данном разделе приводится постановка трех NP-полных (в обычном смысле) задач разбиения (РАЗБИЕНИЕ, ЧЁТНО–НЕЧЁТНОЕ РАЗБИЕНИЕ, ЧЁТНО–НЕЧЁТНОЕ РАЗБИЕНИЕ С ОГРАНИЧЕНИЯМИ), и приводятся схемы полиномиального сведения этих задач друг к другу и к задаче $1 \,||\, \sum T_j$.

Задача РАЗБИЕНИЕ: задано множество из $\bar{n}$ целых положительных чисел $E = \{e_1, e_2, \ldots, e_{\bar{n}}\}$. Требуется определить, существует ли разбиение множе-

ства E на два подмножества E_1 и E_2 такое, что $\sum\limits_{e_i\in E_1} e_i = \sum\limits_{e_i\in E_2} e_i$, $E = E_1 \bigcup E_2$, $E_1 \bigcap E_2 = \emptyset$.

Задача ЧЁТНО-НЕЧЁТНОЕ РАЗБИЕНИЕ (ЧНР): задано упорядоченное множество из $2\bar{n}$ положительных целых чисел $B = \{b_1, b_2, \ldots, b_{2\bar{n}}\}$, $b_i > b_{i+1}$, $1 \le i < 2\bar{n}$. Требуется определить, существует ли разбиение множества B на два подмножества B_1 и B_2, такое что $\sum\limits_{b_i\in B_1} b_i = \sum\limits_{b_i\in B_2} b_i$, и для каждого $i = 1, 2, \ldots, \bar{n}$ подмножество B_1 (следовательно и B_2) содержит в точности один элемент из пары $\{b_{2i-1}, b_{2i}\}$, $B = B_1 \bigcup B_2$, $B_1 \bigcap B_2 = \emptyset$.

Задача ЧЁТНО-НЕЧЁТНОЕ РАЗБИЕНИЕ С ОГРАНИЧЕНИЯМИ (ОЧНР): задано упорядоченное множество из $2\bar{n}$ положительных целых чисел $A = \{a_1, a_2, \ldots, a_{2\bar{n}}\}$, $a_i > a_{i+1}$, $1 \le i < 2\bar{n}$, такое что $a_{2j} > a_{2j+1} + \delta$ для всех $1 \le j < n$ и $a_i > \bar{n}(4\bar{n}+1)\delta + 5\bar{n}(a_1 - a_{2\bar{n}})$ для всех $1 \le i \le 2\bar{n}$, где $\delta = \frac{1}{2}\sum\limits_{i=1}^{\bar{n}}(a_{2i-1} - a_{2i})$. Требуется определить, существует ли разбиение множества A на два подмножества A_1 и A_2 такое, что $\sum\limits_{a_i\in A_1} a_i = \sum\limits_{a_i\in A_2} a_i$, и для каждого $i = 1, 2, \ldots, \bar{n}$ подмножество A_1 (следовательно и A_2) содержит в точности один элемент из пары $\{a_{2i-1}, a_{2i}\}$, $A = A_1 \bigcup A_2$, $A_1 \bigcap A_2 = \emptyset$.

Задача РАЗБИЕНИЕ является NP-трудной в обычном смысле, известен псевдополиномиальный алгоритм решения данной задачи трудоёмкостью $O(\bar{n}\sum e_i)$ операций [7].

Покажем, что задача ЧНР является NP-трудной в обычном смысле, приведём схему полиномиального сведения примеров задачи РАЗБИЕНИЕ к примерам задачи ЧНР. Пусть $E = \{e_1, e_2, \ldots, e_{\bar{n}}\}$ – произвольный пример задачи РАЗБИЕНИЕ, построим пример задачи ЧНР $B = \{b_1, b_2, \ldots, b_{2\bar{n}}\}$ следующим образом. Положим $b_{2\bar{n}} = 1$, $b_{2\bar{n}-1} = e_{\bar{n}} + 1$, а также

$$\begin{cases} b_{2i} = b_{2i+1} + 1, \\ b_{2i-1} = b_{2i} + e_i, \end{cases} \qquad i = \bar{n}-1, \bar{n}-2, \ldots, 1.$$

Отметим, что выполняется

$$b_{2i-1} - b_{2i} = e_i, \qquad i = 1, \ldots, \bar{n}. \tag{7.2}$$

Пусть E_1 и E_2 является решением примера задачи РАЗБИЕНИЕ. Положим $B_1 = \{b_{2i-1} : e_i \in E_1\} \bigcup \{b_{2i} : e_i \in E_2\}$ и $B_2 = \{b_{2i-1} : e_i \in E_2\} \bigcup \{b_{2i} : e_i \in E_1\}$. При этом, подмножество B_1 (следовательно и B_2) содержит в точности один элемент из каждой пары $\{b_{2i-1}, b_{2i}\}$, $i = 1, \ldots, \bar{n}$. Из равенств $\sum\limits_{e_i\in E_1} e_i = \sum\limits_{e_i\in E_2} e_i$ и (7.2) следует $\sum\limits_{b_i\in B_1} b_i = \sum\limits_{b_i\in B_2} b_i$, т.е. разбиение B_1 и B_2 является решением построенного примера задачи ЧНР.

И наоборот, если B_1 и B_2 является решением примера задачи ЧНР, то положим $E_1 = \{e_i : b_{2i-1} \in B_1\}$ и $E_2 = \{e_i : b_{2i-1} \in B_2\}$. При этом, из равенств $\sum\limits_{b_i \in B_1} b_i = \sum\limits_{b_i \in B_2} b_i$ и (7.2) следует $\sum\limits_{e_i \in E_1} e_i = \sum\limits_{e_i \in E_2} e_i$, т.е. разбиение E_1 и E_2 является решением исходного примера задачи РАЗБИЕНИЕ. Следовательно, задача ЧНР является NP-трудной в обычном смысле.

Лемма 7.1 [114] *Задача ОЧНР является NP-трудной в обычном смысле.*

Доказательство. Доказательство леммы 7.1 основывается на полиномиальной схеме сведения примеров задачи ЧНР к задаче ОЧНР. Пусть $B = \{b_1, b_2, \ldots, b_{2n}\}$ —пример задачи ЧНР. Построим пример задачи ОЧНР $A = \{a_1, a_2, \ldots, a_{2n}\}$ следующим образом:

$$a_j = b_j + (9\bar{n}^2 + 3\bar{n} - i + 1)\Delta + 5\bar{n}(b_1 - b_{2\bar{n}}),\ j = 1, 2, \ldots, 2\bar{n},$$

где $\Delta = \frac{1}{2}\sum\limits_{i=1}^{\bar{n}}(b_{2i-1} - b_{2i})$. Так как $a_{2i-1} - a_{2i} = b_{2i-1} - b_{2i}$, $1 \le i \le 2\bar{n}$, то выполняется $\Delta = \delta$.

Пусть B_1 и B_2 является решением примера задачи ЧНР, т.е. выполняется $\sum\limits_{b_i \in B_1} b_i = \sum\limits_{b_i \in B_2} b_i$ и подмножество B_1 содержит в точности один элемент их каждой пары $\{b_{2i-1}, b_{2i}\}$, $1 \le i \le 2\bar{n}$. Пусть $A_1 = \{a_i : b_i \in B_1\}$ и $A_2 = \{a_i : b_i \in B_2\}$. При этом выполняется $\sum\limits_{a_i \in a_1} a_i = \sum\limits_{a_i \in A_2} a_i$ и подмножество A_1 содержит в точности один элемент их каждой пары $\{a_{2i-1}, a_{2i}\}$, $1 \le i \le 2\bar{n}$. Следовательно, разбиение A_1, A_2 является решением примера задачи ОЧНР. И наоборот, если A_1, A_2 является решением примера задачи ОЧНР, то положим $B_1 = \{b_i : a_i \in A_1\}$ и $B_2 = \{b_i : a_i \in A_2\}$. При этом выполняется $\sum\limits_{b_i \in B_1} b_i = \sum\limits_{b_i \in B_2} b_i$, и подмножество B_1 содержит в точности один элемент из каждой пары $\{b_{2i-1}, b_{2i}\}$, $1 \le i \le 2\bar{n}$. Следовательно, разбиение B_1, B_2 является решением примера задачи ЧНР. Таким образом, пример A имеет решение “Да” тогда и только тогда, когда пример B имеет решение “Да”. □

Без потери общности будем предполагать, что $a_{2i-1} \le a_{2i} + \delta$ для каждого $1 \le i \le \bar{n}$, в противном случае решения примера задачи ОЧНР не существует.

Построим пример задачи $1||\sum T_j$ с множеством из $n = 3\bar{n} + 1$ требований $N = \{V_1, V_2, \ldots, V_{2\bar{n}}\} \bigcup \{W_1, W_2, \ldots, W_{\bar{n}+1}\}$ и моментом начала обслуживания $t_0 = 0$. Первые $2\bar{n}$ требований множества N будем называть

V-требованиями, а остальные $\bar{n}+1$ требования — W-требованиями. Пусть $b=(4\bar{n}+1)\delta$. Зададим параметры требований следующим образом:

$$
\begin{aligned}
&p_{V_i}{=}a_i, \qquad 1\le i\le 2\bar{n};\\
&p_{W_i}{=}b, \qquad 1\le i\le \bar{n}+1;\\
&d_{V_i}{=}\begin{cases}(j-1)b+\delta+(a_2+a_4+\ldots+a_{2j}), & \text{если } i=2j-1,\\ d_{V_{2j-1}}+2(\bar{n}-j+1)(a_{2j-1}-a_{2j}), & \text{если } i=2j;\end{cases}\\
&d_{W_i}{=}\begin{cases}ib+(a_2+a_4+\ldots+a_{2i}), & 1\le i\le\bar{n},\\ d_{W_{\bar{n}}}+\delta+b, & i=\bar{n}+1.\end{cases}
\end{aligned}
$$

Заметим, что W-требования имеют одинаковую продолжительность, и каждое из V-требований соответствует элементам множества A.

Пусть множества $\{V_{1,1},V_{2,1},\ldots,V_{\bar{n},1}\}$ и $\{V_{1,2},V_{2,2},\ldots,V_{\bar{n},2}\}$ являются разбиением множества V-требований так, что $\{V_{i,1},V_{i,2}\}=\{V_{2i-1},V_{2i}\}$ для каждого $1\le i\le n$. Определим *каноническое расписание* как расписание вида

$$\pi=(V_{1,1},W_1,V_{2,1},W_2,\ldots,W_{\bar{n}-1},V_{\bar{n},1},W_{\bar{n}},W_{\bar{n}+1},V_{\bar{n},2},V_{\bar{n}-1,2},\ldots,V_{1,2}). \quad (7.3)$$

Примеры задачи $1||\sum T_j$, к которым сводится примеры задачи ЧНР, будем называть *каноническими примерами.*

Лемма 7.2 [114] *Для всех канонических примеров задачи* $1||\sum T_j$ *существует оптимальное расписание, которое является каноническим.*

□

Пусть $\bar{A}=\frac{1}{2}\sum\limits_{i=1}^{2\bar{n}}a_i$, $C=(\bar{n}+1)b+2\bar{A}$ и
$\omega_0=\bar{A}+\bar{n}C-\bar{n}(\bar{n}-1)b/2-\bar{n}\delta-\sum_{i=1}^{\bar{n}}(\bar{n}-i+1)(a_{2i-1}+a_{2i})$.

Тогда справедлива следующая лемма.

Лемма 7.3 [114] *Если* π *каноническое расписание, то* $F(\pi)\ge\omega_0$. *При этом,* $F(\pi)=\omega_0$ *тогда и только тогда, когда* $\sum\limits_{i=1}^{\bar{n}}p_{V_{i,1}}=\sum\limits_{i=1}^{\bar{n}}p_{V_{i,2}}$. □

Таким образом, пример задачи ЧНР имеет решение “Да” тогда и только тогда, когда оптимальное каноническое расписание π^* соответствующего канонического примера задачи $1||\sum T_j$ будет иметь суммарное запаздывание равное величине ω_0. В этом случае расписание π^* будет однозначно задавать требуемое разбиение множества B на два подмножества $B_1=\{V_{1,1},V_{2,1},\ldots,V_{\bar{n},1}\}$ и $B_2=\{V_{1,2},V_{2,2},\ldots,V_{\bar{n},2}\}$. Если пример задачи ЧНР не имеет решения, то будет выполняться $F(\pi^*)>\omega_0$.

7.2.2 Алгоритм *B*-1-*канонический*

Как было показано в разделе 7.2.1 для канонических примеров задачи $1 \,||\, \sum T_j$ (т.е. примеров, полученных в результате полиномиального сведения примеров задачи ЧЁТНО–НЕЧЁТНОЕ РАЗБИЕНИЕ С ОГРАНИЧЕНИЯМИ) существует каноническое расписание, которое является оптимальным. Каноническое расписание имеет вид (7.3). Учитывая нумерацию требований канонического примера:

$$V_1 < V_2 < \ldots < V_{2\bar{n}} < W_1 < W_2 < \ldots < W_{\bar{n}+1},$$

можно утверждать, что любое, в том числе и оптимальное, каноническое расписание обладает свойством B-1, т.е. для любого канонического примера I выполняется $\Pi^*_{B-1}(I) \neq \emptyset$. Следовательно, согласно теореме 7.1 алгоритм B-1 находит оптимальное расписание для канонических примеров. Это означает, что данный алгоритм может быть использован для решения примеров задач РАЗБИЕНИЕ, определенных в разделе 7.2.1.

Отметим, что для каждой тройки требований V_{2i-1}, V_{2i}, W_i, $i = 1, \ldots, \bar{n}$, любого канонического примера существуют только две возможности их взаимного обслуживания при любом каноническом расписании π, а именно либо $(V_{2i-1} \to W_i \to V_{2i})_\pi$, либо $(V_{2i} \to W_i \to V_{2i-1})_\pi$. Это свойство позволяет модифицировать алгоритм B-1 для решения канонических примеров.

Шаги 1 и 3–8 алгоритма B-1-*канонический* выполняются для каждой целочисленной точки $t \in \left[\sum_{i=1}^{k-1} a_{2i} + (k-1)b, \sum_{i=1}^{k-1} a_{2i} + (k-1)b + 2\delta\right]$, где, напомним, $\delta = \frac{1}{2}\sum_{i=1}^{\bar{n}}(a_{2i-1} - a_{2i})$.

Теорема 7.2 *Алгоритм B-1-канонический находит оптимальное каноническое расписание (и следовательно находит решение задач РАЗБИЕНИЕ, ЧНР и ОЧНР) за $O(\bar{n}\delta)$ операций.*

Алгоритм 7.1 Алгоритм B-1-*канонический*

1: $\pi^*_{\bar{n}}(t) := (W_{\bar{n}+1})$, $F_{\bar{n}}(t) := \max\{0, p_{W_{\bar{n}+1}} - t\}$;
2: **for all** $k = \bar{n}-1, \bar{n}-2, \ldots, 1$ **do**
3: $\pi^1 := (V_{2k-1}, W_k, \pi^*_{k+1}(t - a_{2k-1} - b), W_{2k})$;
4: $\pi^2 := (V_{2k}, W_k, \pi^*_{k+1}(t - a_{2k} - p_{W_k}), V_{2k+1})$;
5: $F(\pi^1) := \max\{0, a_{2k-1} - d_{V_{2k-1}}(t)\} + \max\{0, a_{2k-1} + b - d_{W_k}(t)\} +$
$F_{k+1}(t - a_{2k-1} - b) + \max\{0, \sum_{j=k}^{n}(a_{2j-1} + a_{2j} + b) - d_{V_{2k}}(t)\}$;
6: $F(\pi^2) := \max\{0, a_{2k} - d_{V_{2k}}(t)\} + \max\{0, a_{2k} + b - d_{W_k}(t)\} +$
$F_{k+1}(t - a_{2k} - b) + \max\{0, \sum_{j=k}^{n}(a_{2j-1} + a_{2j} + b) - d_{V_{2k-1}}(t)\}$;
7: $F^*_k(t) := \min\{F(\pi^1), F(\pi^2)\}$;
8: $\pi^*_k(t) := \arg\min\{F(\pi^1), F(\pi^2)\}$;
9: **end for**
10: RETURN расписание $\pi^*_1(d_{W_{\bar{n}+1}})$ и его значение суммарного запаздывания $F_1(d_{W_{\bar{n}+1}})$.

Доказательство. Как было показано выше, алгоритм B-1 находит оптимальное расписание для любого канонического примера. Алгоритм B-1-*канонический* является модификацией данного алгоритма, который позволяет исключить из рассмотрения заведомо неоптимальные варианты обслуживания троек требований V_{2i-1}, V_{2i}, W_i, $i = 1, \ldots, \bar{n}$, поэтому алгоритм B-1-*канонический* находит оптимальное расписание для любого канонического примера (и следовательно, находит решение задач РАЗБИЕНИЕ, ЧНР и ОЧНР).

Оценим трудоёмкость построения расписания $\pi^*_1(d_{W_{\bar{n}+1}})$ рассматриваемым алгоритмом. Для каждого k, $k = \bar{n}-1, \bar{n}-2, \ldots, 1$, достаточно рассматривать только те целочисленные точки t, которые принадлежат отрезку $\left[\sum_{i=1}^{k-1} a_{2i} + (k-1)b, \sum_{i=1}^{k-1} a_{2i} + (k-1)b + 2\delta\right]$, поскольку для точек t вне данного отрезка расписание $\pi^*_k(t)$ не будет каноническим. Длина данного отрезка равна 2δ, для фиксированных t и k шаг алгоритма выполняется за $O(1)$ операций. Следовательно, трудоёмкость алгоритма B-1-*канонический* составляет $O(\bar{n}\delta)$ операций. □

Отметим следующий момент. Рассмотрим произвольный пример задачи РАЗБИЕНИЕ $E = \{e_1, \ldots, e_{\bar{n}}\}$. Применим к данному примеру схемы полиномиального сведения, приведенные в разделе 7.2.1 и последовательно получим конкретные примеры задач ЧНР (B), ОЧНР (A) и канонический пример задачи $1\,||\sum T_j$. Для построенных примеров выполняется

$$\delta = \frac{1}{2}\sum_{i=1}^{\bar{n}}(a_{2i-1} - a_{2i}) = \frac{1}{2}\sum_{i=1}^{\bar{n}}(b_{2i-1} - b_{2i}) = \frac{1}{2}\sum_{i=1}^{\bar{n}} e_i.$$

Следовательно, алгоритм *B*-1-*канонический* находит решение любого примера задачи РАЗБИЕНИЕ за $O(n\sum e_i)$ операций, что не хуже трудоёмкости нахождения решения примеров задачи РАЗБИЕНИЕ с помощью известного алгоритма Гэри и Джонсона [7].

7.3 Алгоритм *B*-1-*модифицированный*

Рассмотрим функцию $f(t)$, обладающую следующими свойствами:

(i) $f(t)$ – непрерывная монотонно-невозрастающая функция, принимающая неотрицательные значения;

(ii) $f(t)$ – кусочно-линейная функция, $\mathbb{T}_1, \mathbb{T}_2, \ldots, \mathbb{T}_{m+1}$ – интервалы линейности функции;

(iii) существует точка $t' \in \mathbb{R}$ такая, что для всех $t \geq t'$ выполяется $f(t) = 0$, для всех $t < t'$ значение функции положительно $f(t) > 0$ и на интервале $(-\infty, t']$ функция $f(t)$ монотонно убывает;

(iv) $f(t-\epsilon) - f(t) \leq n\epsilon$ для любых значений $t \in \mathbb{R}$ и $\epsilon > 0$.

Заметим, что условие (iv) означает, что коэффициент наклона рассматриваемой функции $f(t)$ в любой точке $t \in \mathbb{T}_i$, $i = 1, \ldots, m+1$, по модулю не превышает значения n.

Пусть $\mathbb{T}_1 = (-\infty; t_1]$, $\mathbb{T}_i = (t_{i-1}; t_i]$, $i = 2, \ldots, m$, и $\mathbb{T}_{m+1} = (t_m; +\infty)$. При этом, согласно условию (iii), $f(t) = 0$ для всех $t \in \mathbb{T}_{m+1}$. Через k_i обозначим абсолютную величину коэффициента наклона функции $f(t)$ в точках $t \in \mathbb{T}_i$, $i = 1, \ldots, m+1$. При этом $k_{m+1} = 0$. Вид функции $f(t)$ показан на рис. 7.1. Функция $f(t)$ записывается следующим образом:

$$f(t) = \begin{cases} 0, \text{ для } t \in \mathbb{T}_{m+1}, \\ k_i(t_i - t) + \sum\limits_{j=i+1}^{m} k_j(t_j - t_{j-1}), \text{ для } t \in \mathbb{T}_i, i = 1, \ldots, m. \end{cases} \tag{7.4}$$

Для того чтобы указать принадлежность параметров $m, k_i, t_i, \mathbb{T}_i$ функции f будем использовать запись $m^f, k_i^f, t_i^f, \mathbb{T}_i^f$. Далее, через $\mathcal{KT}_f$ будем обозначать упорядоченное множество пар $\langle k_i^f, t_i^f \rangle$, $i = 1, \ldots, m^f$, и использовать запись $\mathcal{KT}_f = \{\langle k_i^f, t_i^f \rangle\}_{i=1,\ldots,m^f}$. При этом выполняется $t_1^f < t_2^f < \ldots < t_{m^f}^f$.

Для любой функции $f(t)$, удовлетворяющей условиям (i)–(iv), существует соответствующее множество $\mathcal{KT}_f$ такое, что данная функция

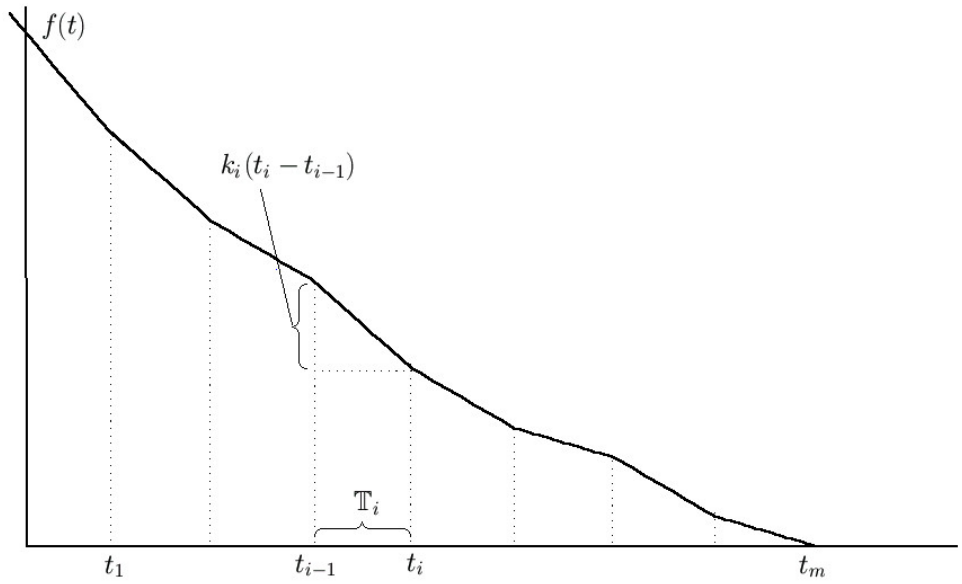

Рис. 7.1: Кусочно-линейная функция $f(t)$.

может быть представлена в виде (7.4). И наоборот, любое множество $\mathcal{KT}_h = \{\langle k_i^f, t_i^f \rangle\}_{i=1,\ldots,m^f}$ задает некоторую кусочно-линейную функцию $f(t)$, удовлетворяющую условиям (i)–(iv). Таким образом, каждую такую функцию $f(t)$ будем представлять некоторым подходящим множеством $\mathcal{KT}_f$.

Рассмотрим следующие операции над двумя функциями $f(t)$ и $g(t)$ вида (i)–(iv).

Операция сдвига функции, т.е. построение такой функции $h(t)$, что $h(t) = f(t + a)$, где a – произвольная константа.

Алгоритм 7.2 Операция сдвига функции

1: *Шаг 1.* $\mathcal{KT}_h := \mathcal{KT}_f$, $m^h := m^f$;
2: *Шаг 2.* Для всех пар $\langle k_i^h, t_i^h \rangle$ из множества $\mathcal{KT}_h$ выполнить $t_i^h := t_i^h + a$.

Операция сложения двух функций , т.е. построение такой функции $h(t)$, что $h(t) = f(t) + g(t)$.

Операция нахождения функции минимума из двух функций , т.е. построение такой функции $h(t)$, что $h(t) = \min\{f(t), g(t)\}$. Каждому интервалу линейности $\mathbb{T}_i^h$, $i = 1, \ldots, m^h$, функции $h(t)$ (т.е. каждой паре $\langle k_i^h, t_i^h \rangle \in \mathcal{KT}_h$) поставим в соответствие величину σ_i^h такую, что $\sigma_i^h = 0$, если для всех $t \in \mathbb{T}_i^h$ $h(t) = f(t)$, и $\sigma_i^h = 1$, если для всех $t \in \mathbb{T}_i^h$ выполняется $h(t) \neq f(t)$. При этом, ситуацию, когда для одних $t \in \mathbb{T}_i^h$ выполняется $h(t) = f(t)$ и для других $t \in \mathbb{T}_i^h$ выполняется $h(t) \neq f(t)$ будем исключать, т.е. разбивать каждый

Алгоритм 7.3 Операция сложения двух функций

1: *Шаг 1.* $\mathcal{KT}_h := \mathcal{KT}_f$, $m^h := m^f$;

2: *Шаг 2.* Для каждой пары $\langle k_i^g, t_i^g \rangle \in \mathcal{KT}_g$, $i = 1, \ldots, m^g$, выполнить:

2.1 if $t_i^g > t_j^h$ for all $j = 1, \ldots, m^h$, then

· добавить в множество $\mathcal{KT}_h$ пару $\langle k_i^g, t_i^g \rangle$;

2.2 else

· найти пару $\langle k', t' \rangle \in \mathcal{KT}_h$ такую, что $t' = \min\limits_{j=1,\ldots,m^h} \{t_j^h : t_j^h \geq t_i^g\}$;

· if $t' \neq t_i^g$, then добавить в множество $\mathcal{KT}_h$ пару $\langle k', t_i^g \rangle$;

2.3 для всех пар $\langle k_j^h, t_j^h \rangle$ из множества $\mathcal{KT}_h$ таких, что $t_j^h \leq t_i^g$, выполнить $k_j^h := k_j^h + k_i^g$;

3: *Шаг 3.* Упорядочить элементы множества $\mathcal{KT}_h$ в порядке возрастания величин t_i^h.

интервал линейности на подинтервалы до тех пор, пока данное условие не выполнится. Также, будем предполагать, что $\sigma_{m^h+1}^h = \sigma_{m^h}^h$.

Поскольку интервалы линейности каждой из функций $f(t)$ и $g(t)$ задают разбиение числовой оси, то построенные интервалы также будут разбиением числовой оси, т.е. $\bigcup\limits_{i,j} \mathbb{T}_{ij} = \mathbb{R}$ и $\mathbb{T}_{ij} \bigcap \mathbb{T}_{i'j'} = \emptyset$ для одновременно $i \neq i'$ и $j \neq j'$. В пункте 2.1 алгоритма 7.4 проверяется существует ли решение уравнения $f(t) = g(t)$ на интервале $\mathbb{T}_{ij}$. Иными словами, необходимо найти точку t_{ij}, которая является решением уравнения

$$k_i^f(t_i^f - t) + \sum_{l=i+1}^{m^f} k_l^f(t_l^f - t_{l-1}^f) = k_j^g(t_j^g - t) + \sum_{l=j+1}^{m^g} k_l^g(t_l^g - t_{l-1}^g)$$

относительно неизвестной t, т.е.

$$t_{ij} = \frac{\sum\limits_{l=i+1}^{m^f} k_l^f(t_l^f - t_{l-1}^f) - \sum\limits_{l=j+1}^{m^g} k_l^g(t_l^g - t_{l-1}^g) + k_i^f t_i^f - k_j^g t_j^g}{k_i^f - k_j^g}.$$

Алгоритм 7.4 Нахождения функции минимума из двух функций

1: *Шаг 1.* Находим интервалы $\mathbb{T}_{ij} := \mathbb{T}_i^f \bigcap \mathbb{T}_j^g$ для каждого $i = 1, \ldots, m^f$ и $j = 1, \ldots, m^g$.

2: *Шаг 2.* Для каждого интервала $\mathbb{T}_{ij} \neq \emptyset$ $i = 1, \ldots, m^f$, $j = 1, \ldots, m^g$ выполнить:

2.1 определить, существует ли решение уравнения $f(t) = g(t)$ на интервале $\mathbb{T}_{ij}$.

2.2 if $t_{ij} \in \mathbb{T}_{ij}$, then:

· if $k_i^f < k_j^g$, then добавим в множество $\mathcal{KT}_h$ пару $\langle k_i^f, t_{ij} \rangle$ и сопоставим этой паре величину $\sigma^h = 0$; else добавим пару $\langle k_j^g, t_{ij} \rangle$ и сопоставим этой паре величину $\sigma^h = 1$.

3: *Шаг 3.* if $t_{m^f} < t_{m^g}$ (or $t_{m^f} = t_{m^g}$ and $k_{m^f} < k_{m^g}$), then

· добавим в множество $\mathcal{KT}_h$ пару $\langle k_{m^f}^f, t_{m^f}^f \rangle$ и сопоставим ей величину $\sigma^h = 0$;

else

· добавим в множество $\mathcal{KT}_h$ пару $\langle k_{m^g}^g, t_{m^g}^g \rangle$ и сопоставим ей величину $\sigma^h = 1$.

4: *Шаг 4.* Упорядочим множество $\mathcal{KT}_h$ по возрастанию величин t_i^h.

5: *Шаг 5.* Пусть $\mathcal{KT}_{\bar{h}} := \mathcal{KT}_h$. Для любых двух соседних элементов $\langle k_i^{\bar{h}}, t_i^{\bar{h}} \rangle$ и $\langle k_{i+1}^{\bar{h}}, t_{i+1}^{\bar{h}} \rangle$, $i = 1, \ldots, m^{\bar{h}} - 1$ множества $\mathcal{KT}_{\bar{h}}$ выполнить:

5.1 if $\sigma_{i+1}^{\bar{h}} = 0$, then

· добавить в множество $\mathcal{KT}_h$ все пары $\langle k_j^f, t_j^f \rangle \in \mathcal{KT}_f$, для которых $t_i^{\bar{h}} < t_j^f < t_{i+1}^{\bar{h}}$ и сопоставить им величину $\sigma^h = 0$;

else

· добавить в множество $\mathcal{KT}_h$ все пары $\langle k_j^g, t_j^g \rangle \in \mathcal{KT}_g$, для которых $t_i^{\bar{h}} < t_j^g < t_{i+1}^{\bar{h}}$ и сопоставить им величину $\sigma^h = 1$;

5.2 if $\sigma_1^{\bar{h}} = 0$, then

· добавить в множество $\mathcal{KT}_h$ все пары $\langle k_j^f, t_j^f \rangle \in \mathcal{KT}_f$, для которых $t_j^f < t_1^{\bar{h}}$ и сопоставить им величину $\sigma^h = 0$;

else

· добавить в множество $\mathcal{KT}_h$ все пары $\langle k_j^g, t_j^g \rangle \in \mathcal{KT}_g$, для которых $t_j^g < t_1^{\bar{h}}$ и сопоставить им величину $\sigma^h = 1$;

6: *Шаг 6.* Упорядочить элементы множества $\mathcal{KT}_h$ в порядке возрастания величин t_i^h.

В случае, если $k_i^f - k_j^g = 0$ (т.е. графики функций $f(t)$ и $g(t)$ на интервале $\mathbb{T}_{ij}$ совпадают или параллельны) считаем, что решения уравнения не

существует (полагаем, например, что $t_{ij} = +\infty$).

На шаге 4 происходит упорядочивание множества $\mathcal{KT}_h$ по возрастанию величин t_i^h. Заметим, что для любых двух соседних элементов множества $\mathcal{KT}_h$ выполняется $\sigma_i^h \neq \sigma_{i+1}^h$, $i = 1, \ldots, m^h$. Это следует из непрерывности рассматриваемых функций.

Отметим, что функции $h(t)$, полученные с помощью вышеуказанных операций, также удовлетворяют условиям (i)–(iv). Трудоёмкость применения рассматриваемых операций полиномиальным образом зависит от величины $m = \max\{m^f, m^g\}$ (можно привести алгоритмические реализации данных операций, трудоёмкость которых не превышает $O(m^2)$ операций).

Через $F_k^\pi(t)$, $k = 1, \ldots, n$, $t \in \mathbb{R}$, будем обозначать значение целевой функции при фиксированном расписании $\pi \in \Pi(I_k(t))$ для примера $I_k(t) = \langle \{p_j, d_j(t)\}_{j \in N_k}, 0 \rangle$, где $N_k = \{k, \ldots, n\}$ (при этом предполагается, что $\{\pi\} = N_k$). В силу регулярности критерия суммарного запаздывания $F_k^\pi(t)$ является невозрастающей функцией.

Пусть $\pi = (j_k, j_{k+1}, \ldots, j_n)$. Определим точки τ_i, $i = k, \ldots, n$, как решение уравнения $C_{j_i}(\pi) = d_{j_i}(t)$, т.е. $\tau_i = C_{j_i}(\pi) - d_{j_i} + d_n + t_0$. Таким образом, точка τ_i, $i = k, \ldots, n$, определяет такой момент времени, что для $t < \tau_i$ ($t \geq \tau_i$) требование j_i запаздывает (не запаздывает) при фиксированном расписании π для примера $I_k(t)$.

Упорядочим множество $\{\tau_k, \ldots, \tau_n\}$ по невозрастанию величин и получим множество $\{\tau'_k, \ldots, \tau'_n\}$, $\tau'_k \geq \tau'_{k+1} \geq \ldots \geq \tau'_n$. Для $t \geq \tau'_n$ при расписании π не запаздывает ни одно требование и, следовательно, $F_k^\pi(t) = 0$. Для $t < \tau'_k$ при расписании π запаздывают все требования и $F_k^\pi(t)$ является линейной функцией с коэффициентом наклона (тангенс угла наклона) равным по модулю $n-k+1$. Для двух соседних точек τ'_{i-1} и τ'_i, $i = k+1, \ldots, n$, выполняется: на интервале $(\tau'_{i-1}; \tau'_i]$ функция $F_k^\pi(t)$ является линейной функцией с коэффициентом наклона равным по модулю количеству запаздывающих требований при расписании π. Таким образом, функция $F_k^\pi(t)$ удовлетворяет условиям (i)–(iv).

Рассмотрим некоторое подмножество расписаний $\Pi \subseteq \Pi(I_k(t))$. Определим функцию $F_k^\Pi(t) = \min\limits_{\pi \in \Pi} F_k^\pi(t)$. Данная функция, являясь функцией минимума из функций, удовлетворяющих условиям (i)–(iv), также удовлетворяет этим условиям.

Рассмотрим шаг алгоритма B-1 для некоторого фиксированного $k \in \{1, \ldots, n-1\}$. Для каждой точки t на рассматриваемом шаге построение оптимального расписания $\pi_k^*(t)$ для примера $I_k(t)$ происходит следующим об-

разом. Строятся два вспомогательных расписания $\pi^1(t) = (k, \pi^*_{k+1}(t - p_k))$ и $\pi^2(t) = (\pi^*_{k+1}(t), k)$ (при формальном описании алгоритма B-1 в разделе 6.3.1 данные расписания обозначались просто π^1 и π^2) и выбирается наилучшее из них. Пусть $F^1(t)$ и $F^2(t)$ будут значениями суммарного запаздывания при данных расписаниях в зависимости от значения t. Так как расписания $\pi^1(t)$ и $\pi^2(t)$ являются наилучшими расписаниями для примера $I_k(t)$, при которых требование k обслуживается на, соответственно, первой и последней позиции, то функции $F^1(t)$ и $F^2(t)$ являются функциями минимума для некоторых подмножеств расписаний. Следовательно, данные функции, равно как и функция $F^*_k(t)$ удовлетворяют условиям (i)–(iv).

Таким образом, на основе введенных выше в данном параграфе понятий и операций над кусочно-линейными функциями вида (i)–(iv) процесс вычисления функции $F^*_k(t)$ можно представить следующим образом. Для упрощения записи будем пользоваться следующими обозначениями. Пусть $f(t) = F^*_{k+1}(t)$, $g^1(t) = F^1(t)$, $g^2(t) = F^2(t)$ и $F(t) = F^*_k(t)$. Предполагаем, что для функции $f(t)$ известно множество $\mathcal{KT}_f$. Построим функции $f^1(t)$ и $f^2(t)$ такие, что

$$\mathcal{KT}_{f^1} = \{\langle 1, p_k - d_k + d_n + t_0\rangle\},$$
$$\mathcal{KT}_{f^2} = \{\langle 1, \sum_{j=k}^{n} p_j - d_k + d_n + t_0\rangle\}.$$

Согласно строкам 5 и 6 алгоритма B-1 функции $g^1(t)$ и $g^2(t)$ вычисляются с помощью определенных выше операций следующим образом:

$$g^1(t) = f(t - p_k) + f^1(t), \qquad g^2(t) = f(t) + f^2(t).$$

Тогда $F(t) = \min\{g^1(t), g^2(t)\}$.

Заметим, что при применении *операции нахождения функции минимума* для функций $g^1(t)$ и $g^2(t)$ мы сохраняем информацию о том, значение какой из этих двух функций меньше в данной точке t (с помощью сопоставления каждому интервалу $\mathbb{T}^F_i$, $i = 1, \ldots, m^F$, линейности функции $F(t)$ значения $\sigma^F_i \in \{0, 1\}$). Следовательно, по завершению вычисления функции $F^*_1(t)$ мы можем построить соответствующее расписание $\pi^*_1(t)$ однозначным образом с помощью анализа полученных значений $\sigma^{F^*_k}_i$, $i = 1, \ldots, m^{F^*_k}$, $k = 1, \ldots, n-1$. Таким образом, процесс нахождения расписания $\pi^*_1(d_n)$ алгоритмом B-1 можно рассматривать как процесс построения функции $F^*_1(t)$ (т.е. построения множества $\mathcal{KT}_{F^*_1}$).

Приведём формальное описание алгоритма *B-1-модифицированный*, основная идея которого была изложена выше.

Алгоритм 7.5 Алгоритм *B*-1-*модифицированный*

1: $\mathcal{KT}_{F_n^*} = \{\langle 1, p_n + t_0\rangle\}$;
2: **for all** $k = n-1, n-2, \ldots, 1$ **do**
3: $\quad \mathcal{KT}_{f^1} = \{\langle 1, p_k - d_k + d_n + t_0\rangle\}$, $\mathcal{KT}_{f^2} = \{\langle 1, \sum\limits_{j=k}^{n} p_j - d_k + d_n + t_0\rangle\}$;
4: $\quad g^1(t) := F_{k+1}^*(t - p_k) + f^1(t)$, $g^2(t) := F_{k+1}^*(t) + f^2(t)$; $F_k^*(t) := \min\{g^1(t), g^2(t)\}$;
5: **end for**
6: построить расписание $\pi_1^*(d_n)$ на основе величин $\sigma_i^{F_k^*}$, $i = 1, \ldots, m^{F_k^*}$, $k = 1, \ldots, n-1$.

Приведём формальное описание процедуры построения расписания $\pi = \pi_1^*(d_n)$ на основе величин $\sigma_i^{F_k^*}$, $i = 1, \ldots, m^{F_k^*}$, $k = 1, \ldots, n-1$.

Пусть $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_{n-1})$ – полученные значения $\sigma_i^{F_k^*}$, соответствующие значению функции $F_1^*(t)$ в точке $t = d_n$, т.е.

$$\begin{aligned}
\sigma_1 &:= \sigma_i^{F_1^*}, \text{ где } i \text{ таково, что } d_n \in \mathbb{T}_i^{F_1^*};\\
\sigma_2 &:= \sigma_i^{F_2^*}, \text{ где } i \text{ таково, что } (d_n - \sigma_1 p_1) \in \mathbb{T}_i^{F_2^*};\\
&\ldots\\
\sigma_k &:= \sigma_i^{F_k^*}, \text{ где } i \text{ таково, что } (d_n - \sum_{j=1}^{k-1} \sigma_j p_j) \in \mathbb{T}_i^{F_k^*};\\
&\ldots\\
\sigma_{n-1} &:= \sigma_i^{F_{n-1}^*}, \text{ где } i \text{ таково, что } (d_n - \sum_{j=1}^{n-2} \sigma_j p_j) \in \mathbb{T}_i^{F_{n-1}^*}.
\end{aligned}$$

Пусть $\pi = (j_1, j_2, \ldots, j_n)$. Вычислим величины $pos(k)$, $k = 1, \ldots, n$, которые равны номеру позиции требования k при расписании π, т.е. $j_{pos(k)} := k$.

Алгоритм 7.6 Алгоритм *построения кусочно-линейной функции*

1: $z_n := \sum\limits_{i=1}^{n-1}(1 - \sigma_i)$, $pos(n) := z_n + 1$;
2: **for** $k = n-1, n-2, \ldots, 1$ **do**
3: $\quad z_k := \sum\limits_{i=1}^{n-1}(1 - \sigma_i)$;
4: $\quad pos(k) := z_k + \sigma_k(n-k) + 1$
5: **end for**

Тогда требование $k \in N$ обслуживается при расписании π $pos(k)$-ым по порядку.

Описанная выше схема (алгоритм 7.6) построения кусочно-линейных функций в ходе выполнения алгоритма *B*-1-*модифицированный* может быть применена и для алгоритма *B*-1-*канонический*. Трудоёмкость выполнения алгоритма *B*-1-*модифицированный* полиномиальным образом зависит от максимального количества $\min\limits_{k=1,\ldots,n} F_k^*$ интервалов линейности функций F_k^*.

В заключение отметим, что трудоёмкость выполнения алгоритма B-1-*модифицированный* не изменяется при масштабировании примера в M раз (то есть умножения всех параметров примера на некоторую большую константу M), поскольку количество интервалов линейности функций F_k^*, $k = 1, \ldots, n$, не изменяется при выполнении масштабирования. Тогда как трудоёмкость алгоритма B-1 увеличивается в M раз при таком масштабировании. Также для выполнения алгоритма B-1-*модифицированный* не требуется ограничение на целочисленность параметров задачи, что позволяет решать задачи РАЗБИЕНИЕ с нецелочисленными значениями. (В скобках заметим, что параметры могут быть и отрицательными ...)

Глава 8

Графический подход к решению задач комбинаторной оптимизации

В известных работах [59, 137] рассматриваются классические NP-трудные в обычном смысле задачи комбинаторной оптимизации: **РАЗБИЕНИЕ** и **РАНЕЦ**. Здесь будем рассматривать следующие их формулировки.

Задача РАЗБИЕНИЕ (partition). Задано упорядоченное множество из n положительных целых чисел $B = \{b_1, \ldots, b_n\}$, $b_1 \geq \ldots \geq b_n$. Требуется разбить множество B на два подмножества B_1 и B_2 так, чтобы минимизировать значение:

$$|\sum_{b_i \in B_1} b_i - \sum_{b_i \in B_2} b_i| \longrightarrow \min\,. \tag{8.1}$$

Задача об одномерном РАНЦЕ (knapsack). Задачу можно представить в виде задачи булевого линейного программирования:

$$\begin{cases} f(x) = \sum\limits_{i:=1}^{n} p_i x_i \longrightarrow \max \\ \sum\limits_{i:=1}^{n} w_i x_i \leq C, \\ x_i \in \{0,1\}, i = 1, \ldots, n. \end{cases} \tag{8.2}$$

Когда $p_i = w_i = b_i$, $i = 1, \ldots, n$, и $C = \frac{1}{2}\sum\limits_{j=1}^{n} b_j$, то задачи (8.1) и (8.2) являются эквивалентными.

8.1 Графический алгоритм решения задачи РАЗБИЕНИЕ

В алгоритме последовательно на шаге $\alpha = 1, \ldots, n$ рассматриваются следующие функции:

$$F_\alpha^1(t) = |\sum_{b_j \in B_1(t-b_\alpha)} b_j - \sum_{b_j \in B_2(t-b_\alpha)} b_j|;$$

$$F_\alpha^2(t) = |\sum_{b_j \in B_1(t+b_\alpha)} b_j - \sum_{b_j \in B_2(t+b_\alpha)} b_j|.$$

В функции $B_1(t)$ (аналогично в $B_2(t)$) ставится в соответствие каждой точке t из рассматриваемого интервала $[-\sum\limits_{j:=1}^{\alpha} b_j, \sum\limits_{j:=1}^{\alpha} b_j]$ некоторое множество элементов $\overline{B_1}$ (соответственно, $\overline{B_2}$). Стоит отметить, что на каждом шаге α алгоритма $B_1(t) \bigcup B_2(t) = \{b_1, \ldots, b_{\alpha-1}\} = \overline{B_1} \bigcup \overline{B_2}, \ \forall t$.

8.1.1 Идея графического алгоритма

Последовательно выбирается "включение" очередного числа b_α, где $\alpha = 1, \ldots, n$, в множество $B_1(t)$ или $B_2(t)$. В каждой точке t "включение" числа b_α выбирается так, чтобы минимизировать значение функции $|\sum\limits_{b_j \in B_1(t)} b_j + t - \sum\limits_{b_j \in B_2(t)} b_j|$, где $t = t_1 - t_2$, t_1 – величина, которая будет добавлена к $B_1(t)$, t_2 – к $B_2(t)$. Параметр t характеризует сумму чисел, которая будет добавлена к $B_1(t)$ на последующих шагах. Если $t < 0$, то $-t$ равно сумме чисел, которая будет добавлена к $B_2(t)$ на последующих шагах.

На очередном шаге α из функции $F_{\alpha-1}(t) = |\sum\limits_{b_j \in B_1(t)} b_j - \sum\limits_{b_j \in B_2(t)} b_j|$, полученной на предыдущем $\alpha - 1$ шаге, строится функция

$$F_\alpha(t) := \min\{F_{\alpha-1}(t - b_\alpha), F_{\alpha-1}(t + b_\alpha)\} = \min\{F_\alpha^1(t), F_\alpha^2(t)\},$$

где $F_0(t) := 0, \forall t$. Если $F_\alpha^1(t) < F_\alpha^2(t)$, то $B_1(t) := B_1(t - b_\alpha) \bigcup \{b_\alpha\}$, иначе $B_2(t) := B_2(t + b_\alpha) \bigcup \{b_\alpha\}$. Кусочно-линейную функцию $F_\alpha(t)$ можно представить (и хранить) в табличном виде по точкам “излома”: t_0, t_1, …, t_{m_α}. На промежутке $[t_i - \frac{t_i - t_{i-1}}{2}, t_i + \frac{t_{i+1} - t_i}{2})$, $i = 1, \ldots, m_\alpha - 1$, кусочно-линейная функция $F_\alpha(t)$ задаётся уравнением $F_\alpha(t) = |t - t_i|$, то есть график функции пересекает ось t в точке t_i. Каждому временно́му интервалу $[t_i - \frac{t_i - t_{i-1}}{2}, t_i + \frac{t_{i+1} - t_i}{2})$

соответствует некоторое фиксированное разбиение $(\overline{B}_1; \overline{B}_2)$, то есть $B_1(t^1) = B_1(t^2) = \overline{B}_1$, $\forall t^1,\ t^2 \in [t_i - \frac{t_i - t_{i-1}}{2}, t_i + \frac{t_{i+1} - t_i}{2})$ (аналогично и для $B_2(t)$). Пусть на предыдущем шаге $\alpha - 1$ получена функция $F_{\alpha-1}(t)$, заданная таблично. На шаге α рассматриваем две функции $F_{\alpha-1}(t - b_\alpha)$ и $F_{\alpha-1}(t + b_\alpha)$, задающиеся соответственно двумя таблицами в точках "излома" :

$$t_0 - b_\alpha,\ \ldots,\ t_i - b_\alpha,\ \ldots,\ t_{m_{\alpha-1}} - b_\alpha \quad \text{и}$$

$$t_0 + b_\alpha,\ \ldots,\ t_i + b_\alpha,\ \ldots,\ t_{m_{\alpha-1}} + b_\alpha.$$

Сопоставляя графики обеих функций, рассматриваем временны́е промежутки $[t^1, t^2], t^1, t^2 \in \{t_0 - b_\alpha, \ldots, t_{m_{\alpha-1}} - b_\alpha, t_0 + b_\alpha, \ldots, t_{m_{\alpha-1}} + b_\alpha\}$, на которых функция $F_{\alpha-1}(t - b_\alpha)$ (и, соответственно, $F_{\alpha-1}(t + b_\alpha)$) задается одним уравнением кусочно-линейной функции. На данном промежутке кусочно - линейные функции $|t - a|$ и $|t - b|$ пересекаются не более чем в одной точке (или совпадают).

Очевидно, что функция $F_\alpha(t)$ может быть задана таблицей, состоящей из множества точек $\{t_0 - b_\alpha, \ldots, t_{m_{\alpha-1}} - b_\alpha, t_0 + b_\alpha, \ldots, t_{m_{\alpha-1}} + b_\alpha\}$, которое будет упорядочено по неубыванию, часть точек может сократиться. Рассматриваются интервалы $[t^1, t^2] \in [-\sum\limits_{j:=1}^{n} b_j,\ \sum\limits_{j:=1}^{n} b_j]$. Следовательно, функция $F_\alpha(t)$ будет задаваться не более чем $m_\alpha = 2m_{\alpha-1}$ точками.

Решением задачи является разбиение $(B_1(0), B_2(0))$, полученное на последнем шаге $\alpha = n$. Значение целевой функции равно $F_n(0)$.

8.1.2 Сокращение рассматриваемых интервалов

Так как на шаге n требуется вычислить значение целевой функции и найти разбиение лишь в точке $t = 0$, то на шаге $n - 1$ нам достаточно рассчитать значения в точках $t \in [-b_n, b_n]$. Аналогично на шаге $n - 2$ достаточно рассмотреть интервал $[-b_n - b_{n-1}, b_n + b_{n-1}]$ и т.д.

Следовательно, на каждом шаге α достаточно рассматривать интервал $[-\sum\limits_{j:=\alpha+1}^{n} b_j,\ \sum\limits_{j:=\alpha+1}^{n} b_j]$ вместо интервала $[-\sum\limits_{j:=1}^{n} b_j,\ \sum\limits_{j:=1}^{n} b_j]$.

При построении функции $F_\alpha(t)$ рассматриваются только точки $t \in [-\sum\limits_{j:=\alpha+1}^{n} b_j,\ \sum\limits_{j:=\alpha+1}^{n} b_j]$. Чтобы интервал сокращался максимально быстро, необходимо упорядочить b_j по невозрастанию. Если учесть, что функция $F_\alpha(t), \alpha = 1, \ldots, n$, является чётной, то при практической реализации алгоритма достаточно хранить “половину” таблицы.

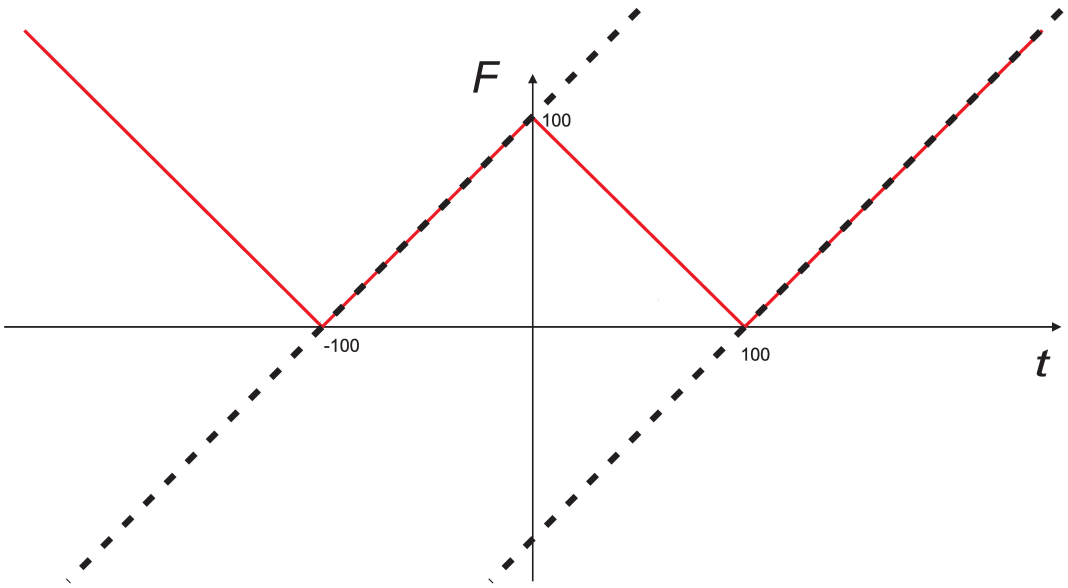

Рис. 8.1: Функция $F_1(t)$.

8.1.3 Пример задачи РАЗБИЕНИЕ

Рассмотрим пример $B = \{100, 70, 50, 20\}, n = 4$. Числа пронумерованы по невозрастанию.

Шаг 0. $F_0(t) := 0,\ B_1(t) = \emptyset, B_2(t) = \emptyset, \forall t.$

Шаг 1. Размещение для $b_1 = 100$. Рассматриваются 2 точки: $0 + 100$ и $0 - 100$. Сравнение функций происходит на трех интервалах $[-240, -100]$, $[-100, 100]$, $[100, 240]$ (или, за счет сокращения интервалов, на $[-140, -100]$, $[-100, 100]$, $[100, 140]$), где $240 = \sum_{j=1}^{n} b_j$.

На интервале $[-240, 0]$ имеем оптимальное разбиение $B_1(t) = \{b_1\}$, $B_2(t) = \emptyset$, на интервале $[0, 240]$ оптимальное разбиение $B_1(t) = \emptyset$, $B_2(t) = \{b_1\}$. Результаты вычислений и целевая функция $F_1(t)$ (в расширенном виде) представлены на рис. 8.1.

Храним следующую информацию:

-100	100
$(100;)$	$(;100)$

Как отмечалось выше, достаточно хранить только “половину” таблицы.

Шаг 2. Размещение для $b_2 = 70$. Рассматриваются 4 точки: $-100-70 = -170;\ -100 + 70 = -30;\ 100 - 70 = 30;\ 100 + 70 = 170$. Вычисления производятся на пяти интервалах $[-240, -170]$, $[-170, -30]$, $[-30, 30]$, $[30, 170]$,

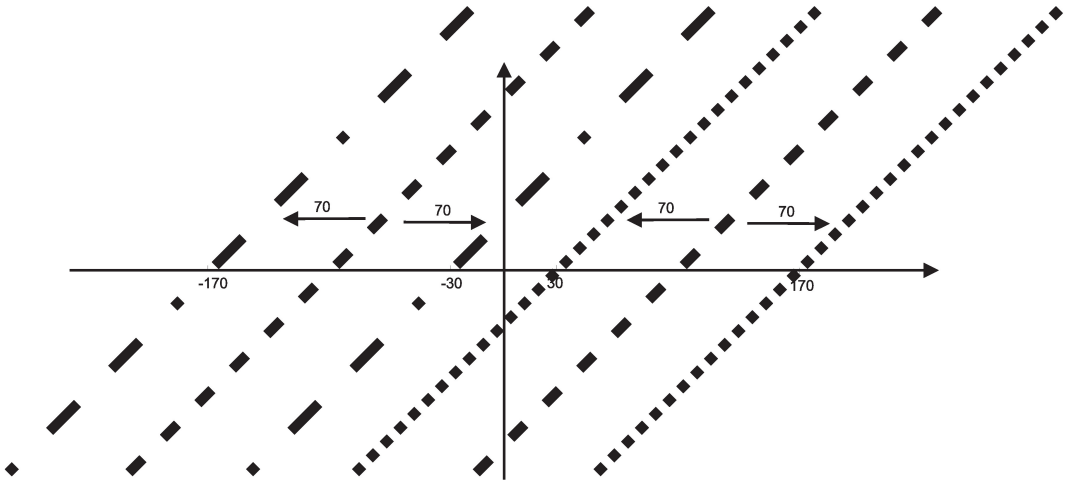

Рис. 8.2: Преобразование функции $F_1(t)$ к функциям $F^1(t)$ и $F^2(t)$.

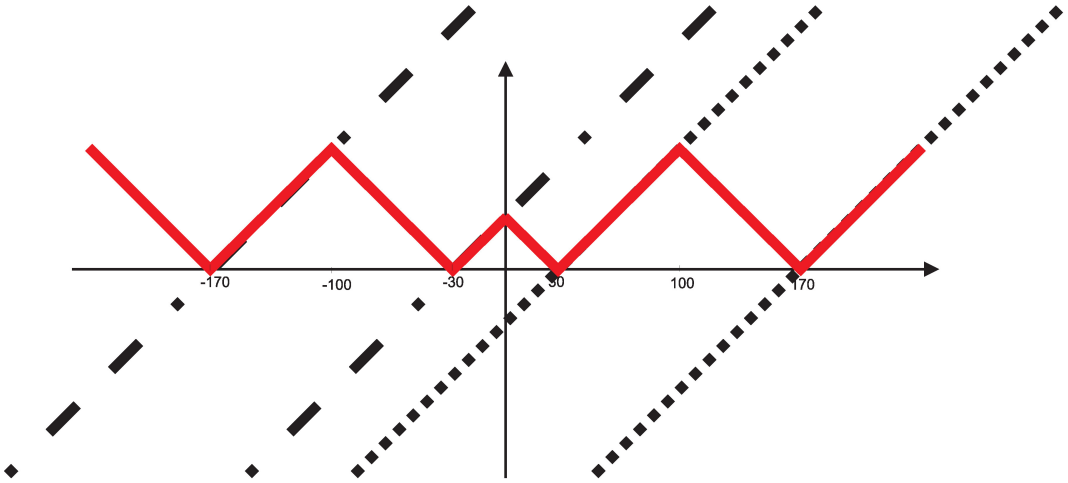

Рис. 8.3: Функция $F_2(t)$.

$[170, 240]$. Но за счёт сокращения, достаточно рассмотреть только три интервала: $[-70, -30], [-30, 30], [30, 70]$. На интервале $[-70, 0]$ имеем оптимальное разбиение $B_1(t) = \{b_1\}, B_2(t) = \{b_2\}$, на интервале $[0, 70]$ оптимальное разбиение $B_1(t) = \{b_2\}, B_2(t) = \{b_1\}$. Фактически не рассматриваем интервалы, а сразу конструируем функцию $F_2(t)$, т.е. включаем в таблицу точки -30 и 30 и соответствующие им разбиения. Точке -30 соответствует разбиение $B_1(t) = \{b_1\}, B_2(t) = \{b_2\}$, а точке 30 – разбиение $B_1(t) = \{b_2\}, B_2(t) = \{b_1\}$. Преобразование функции $F_1(t)$ к функциям $F^1(t)$ и $F^2(t)$ представлено на рис. 8.2.

Результаты вычислений и целевая функция $F_2(t)$ (в расширенном виде) представлены на рис.8.3.

Таким образом, для выполнения следующего шага алгоритма достаточно сохранять информацию только в одной точке $t = 30$:

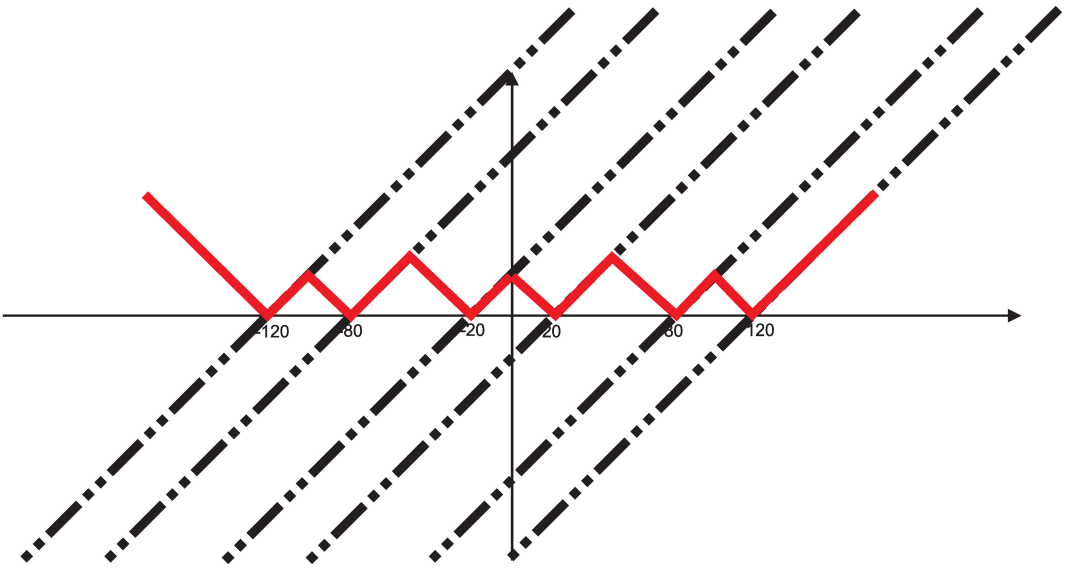

Рис. 8.4: Функция $F_3(t)$.

30
(70; 100)

Шаг 3. Размещение для числа $b_3 = 50$. Рассматриваются четыре точки: $-30-50 = -80$; $-30+50 = 20$; $30+50 = 80$; $30-50 = -20$. За счёт сокращения интервала достаточно рассмотреть один интервал $[-20, 20]$. Результаты вычислений и целевая функция $F_3(t)$ (в расширенном виде) представлены на рис. 8.4. Храним следующую информацию для одной точки $t = 20$:

20
(70, 50; 100)

На шаге 4 мы получаем два оптимальных "симметричных" решения $B_1(0) = \{b_1, b_4\}$, $B_2(0) = \{b_2, b_3\}$ и $B_1(0) = \{b_2, b_3\}$, $B_2(0) = \{b_1, b_4\}$.

Таким образом, рассмотрено семь точек: $2(-100\&100) + 2(-30\&30) + 2(-20\&20) + 1(0) = 7$, в то время как алгоритмом динамического программирования [1] (с сокращением интервалов) пришлось бы просмотреть $280 + 140 + 40 + 0 = 460$ точек. Алгоритм *Balsub* ([137], стр. 83) с трудоёмкостью $O(nb_{\max})$ операций, который является наилучшим из точных алгоритмов решения задачи РАЗБИЕНИЕ, найдет решение за $2 \cdot \frac{n}{2} \cdot b_{\max} = 400$ операций.

Так как функция $F_\alpha(t)$ чётная, то достаточно хранить лишь "половину" таблицы. Кроме того, следует отметить, что при "масштабировании с небольшим изменением параметров" примера, т.е. когда $b'_j = Kb_j + \varepsilon_j$,

где $|\varepsilon_j| \ll K$, $K-$ некоторая достаточно большая положительная константа, $j = 1, \ldots, n$, трудоёмкость алгоритма, построенного на методе динамического программирования, составит $O(Kn\sum b_j)$ операций, в то время как у графического алгоритма трудоёмкость не изменится. Для алгоритма *Balsub* с трудоёмкостью $O(nb_{\max})$ операций такое "масштабирование" примера также приведёт к увеличению трудоёмкости в K раз. Таким образом, графический алгоритм находит решение за одно и тоже количество операций для всех точек некоторого конуса в n-мерном пространстве, если представить параметры примера как точку $(b_1, \ldots, b_n)$ в n-мерном пространстве. Причём параметры примера могут быть как отрицательными так и нецелочисленными.

8.1.4 Трудоёмкость алгоритма

Теорема 8.1 *Графический алгоритм находит на шаге $\alpha = n$ оптимальное разбиение $B_1(0)$ и $B_2(0)$.*

Доказательство. Покажем, что на каждом шаге $\alpha = 1, \ldots, n$ в каждой точке $t \in [-\sum\limits_{j:=\alpha+1}^{n} b_j, \sum\limits_{j:=\alpha+1}^{n} b_j]$ алгоритмом находится оптимальное частичное разбиение $B_1(t)$ и $B_2(t)$ подмножества чисел $\{b_1, \ldots, b_\alpha\}$.

Доказательство проведем по методу математической индукции.

1. Очевидно, что на шаге $\alpha = 1$ в каждой точке $t \in [-\sum\limits_{j:=2}^{n} b_j, \sum\limits_{j:=2}^{n} b_j]$ получаем оптимальное разбиение $B_1(t)$ и $B_2(t)$.

2. Предположим, что на шаге α в каждой точке $t \in [-\sum\limits_{j:=\alpha+1}^{n} b_j, \sum\limits_{j:=\alpha+1}^{n} b_j]$ имеем некоторое оптимальное разбиение $B_1(t)$ и $B_2(t)$.

3. Покажем, что на шаге $\alpha + 1$ в каждой точке $t \in [-\sum\limits_{j:=\alpha+2}^{n} b_j, \sum\limits_{j:=\alpha+2}^{n} b_j]$ алгоритмом будет получено оптимальное разбиение $B_1(t)$ и $B_2(t)$.

   Пусть в точке t алгоритмом рассмотрено два разбиения $(B_1(t-b_{\alpha+1});\ B_2(t-b_{\alpha+1}) \bigcup \{b_{\alpha+1}\})$ и $(B_1(t+b_{\alpha+1}) \bigcup \{b_{\alpha+1}\};\ B_2(t+b_{\alpha+1}))$. В алгоритме выбирается то разбиение, на котором достигается минимальное значение величины $|\sum\limits_{b_j \in B_1} b_j + t - \sum\limits_{b_j \in B_2} b_j|$.

   Пусть в точке t существует такое разбиение $(\overline{B}_1;\ \overline{B}_2)$, что выполняется

$$|\sum_{b_j \in B_1(t+b_{\alpha+1}) \bigcup \{b_{\alpha+1}\}} b_j + t - \sum_{b_j \in B_2(t+b_{\alpha+1})} b_j| > |\sum_{b_j \in \overline{B}_1} b_j + t - \sum_{b_j \in \overline{B}_2} b_j|$$

и

$$|\sum_{b_j \in B_1(t-b_{\alpha+1})} b_j + t - \sum_{b_j \in B_2(t-b_{\alpha+1}) \bigcup \{b_{\alpha+1}\}} b_j| > |\sum_{b_j \in \overline{B}_1} b_j + t - \sum_{b_j \in \overline{B}_2} b_j|.$$

Пусть $b_{\alpha+1} \in \overline{B}_1$, тогда имеем:

$$|\sum_{b_j \in B_1(t+b_\alpha)} b_j + b_{\alpha+1} + t - \sum_{b_j \in B_2(t+b_\alpha)} b_j| > |\sum_{b_j \in \overline{B}_1 \setminus \{b_{\alpha+1}\}} b_j + b_{\alpha+1} + t - \sum_{b_j \in \overline{B}_2} b_j|,$$

но тогда полученное на шаге α в точке $t + b_\alpha$ разбиение $(B_1(t + b_\alpha); B_2(t + b_\alpha))$ не оптимальное, так как разбиение $(\overline{B}_1 \setminus \{b_{\alpha+1}\}; \overline{B}_2)$ "лучше". Получили противоречие. Аналогично может быть показано и для случая $b_{\alpha+1} \in \overline{B}_2$. □

Из анализа графического алгоритма следует:

1. Существует класс целочисленных примеров, на котором количество рассматриваемых точек растёт экспоненциальным образом с ростом n. Например,
   $B = \{b_1, \ldots, b_n\} = \{M, M-1, M-2, \ldots, 1, 1, \ldots, 1\}$. M – достаточно большое число, сумма единиц в примере равна $M(M+1)/2$, т.е. $n = M + M(M+1)/2$.

2. Существует класс нецелочисленных примеров $B = \{b_1, \ldots, b_n\}$, на котором количество рассматриваемых точек также растёт экспоненциальным образом. Например, если не существует такого набора чисел $\lambda_i = \pm 1$, $i = 1, \ldots, n$, что выполняется $\lambda_1 b_1 + \ldots + \lambda_n b_n = 0$, то количество точек в таком примере растет $O(2^n)$.

8.1.5 Экспериментальная оценка трудоёмкости алгоритма

Был проведен сравнительный анализ алгоритмов, представленных в известной работе [137], с предлагаемым графическим алгоритмом. Проведены две группы экспериментов:

1. В первой группе экспериментов для всех целочисленных значений, когда параметры удовлетворяют ограничениям: $40 \geq b_1 \geq \ldots \geq b_n \geq 1$, при $n = 4, 5, \ldots, 10$. Полученные результаты сведены в таблице 8.1.

   В первом столбце – размерность задачи (n); во втором – общее количество решенных примеров для данного n (число сочетаний из $b_{\max} + n - 1$

Таблица 8.1:

1	2	3	4	5	6	7	8	9	10
4	123 410	9	307	328	20	443	640	2	63 684
5	1 086 008	16	444	512	40	564	1000	2	337 077
6	8 145 060	29	542	738	60	687	1440	4	1 140 166
7	53 524 680	48	633	1004	140	811	1960	11	2 799 418
8	314 457 495	76	725	1312	212	933	2560	23	5 348 746
9	1 677 106 640	115	814	1660	376	1053	3240	83	8 488 253
10	8 217 822 536	168	905	2050	500	1172	4000	416	11 426 171

по n, где $b_{\max} = 40$); в третьем – среднее значение трудоёмкости графического алгоритма; в четвертом – среднее значение трудоёмкости алгоритма Balsub; в пятом – среднее значение трудоёмкости алгоритма, основанного на методе динамического программирования; в шестом – максимальное значение трудоёмкости графического алгоритма; в седьмом – максимальное значение трудоёмкости алгоритма Balsub; в восьмом – максимальное значение трудоёмкости алгоритма, основанного на методе динамического программирования; в девятом – количество примеров, для которых трудоёмкость алгоритма Balsub меньше графического алгоритма; в десятом – количество примеров, для которых трудоёмкость алгоритма, основанного на методе динамического программирования меньше алгоритма Balsub.

2. Во-второй группе экспериментов для $n = 4, 5, \ldots, 10$ строились по 20 000 примеров, когда параметры примера выбирались равномерно $b_i \in [1, 200]$, $i = 1, \ldots, n$. Затем для каждого примера в n-мерном пространстве в окрестности $r = 100 + n$ решались $1000n$ примеров $\{b'_1, \ldots, b'_n\}$ таким образом, что $b_i - (100 + n) \le b'_i \le b_i + (100 + n)$, $i = 1, \ldots, n$. Причём, если в окрестности находился пример с большей трудоёмкостью, то "переходим в этот пример". Процесс останавливался, когда в окрестности не удается найти "более сложные примеры". Были получены следующие результаты (см. таблицу 8.2).

   В первом столбце – размерность задачи (n); во втором–четвертом столбцах – среднее значение трудоёмкости алгоритмов: графического, Balsub и алгоритма, основанного на методе динамического программирования в "начальной точке"; в пятом–седьмом столбцах – максимальное значение алгоритмов в "начальной точке"; в 8–9-м – максимальное и среднее количество "переходов от начальной к конечной точке"; в 10–12-м – среднее значение трудоёмкости исследуемых алгоритмов в "конечной точке"; в

Таблица 8.2:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
4	8	1463	1591	16	2196	3080	4	2	6	1310	1604	20	2207	3200	10504	8970
5	16	2191	2490	40	2797	44675	8	3	17	2482	3102	40	2811	5000	6641	3642
6	29	2700	3586	60	3401	6570	12	4	26	2884	3932	60	3418	7176	3000	3170
7	50	3145	4881	140	4006	8729	15	6	54	3353	6794	136	4029	9800	1101	88
8	87	3600	6362	216	4617	11056	19	8	82	3849	7039	220	4645	12112	333	377
9	149	4050	8059	464	5241	13644	52	21	144	4109	11803	476	5232	16200	86	1
10	245	4499	9930	656	5815	16730	24	10	240	4732	12410	676	5854	18840	18	3

13–15-м – максимальное значение трудоёмкости исследуемых алгоритмов в “конечной точке”; в 16–17-м – количество примеров, для которых трудоёмкость алгоритма, основанного на методе динамического программирования меньше трудоёмкости алгоритма Balsub, соответственно в начальных и конечных точках. Во всех (!) примерах (“начальных” и “конечных”) трудоёмкость алгоритма Balsub была больше, чем у графического алгоритма, за исключением 38 примеров для $n = 4$ и двух примеров для $n = 5$ из 20000 “конечных” примеров.

8.2 Графический подход для решения задачи одномерный РАНЕЦ ($0-1$ *knapsack*)

8.2.1 Алгоритм динамического программирования для задачи РАНЕЦ

Наиболее эффективным для задачи считается алгоритм динамического программирования, основанный на принципе оптимальности Беллмана [1], [59]. Алгоритм "работает" только в случае, когда параметры C, $w_i \in Z^+$, $i = 1, \ldots, n$. На каждом шаге $\alpha = 1, \ldots, n$ строится функция

$$g_\alpha(t) = \max_{x_\alpha \in \{0,1\}} \{p_\alpha x_\alpha + g_{\alpha-1}(t - w_\alpha x_\alpha)\}, t \geq w_\alpha x_\alpha,$$

в каждой точке $0 \leq t \leq A$. Для каждой точки t фиксируется соответствующий $x_\alpha = \arg\max g_\alpha(t)$. На шаге $\alpha = n$ в точке $t = C$ находим оптимальное решение.

Продемонстрируем работу алгоритма на примере из [62], стр. 125–129.

$$\begin{cases} f(x) = 5x_1 + 7x_2 + 6x_3 + 3x_4 \longrightarrow \max \\ 2x_1 + 3x_2 + 5x_3 + 7x_4 \leq 9, \\ x_i \in \{0, 1\},\ i = 1, \ldots, 4. \end{cases} \tag{8.3}$$

Работу алгоритма можно представить в виде таблицы 8.3.

Таблица 8.3:

t	$g_1(t)$	$x(t)$	$g_2(t)$	$x(t)$	$g_3(t)$	$x(t)$	$g_4(t)$	$x(t)$
0	0	(0,,,)	0	(0,0,,)	0	(0,0,0,)	0	(0,0,0,0)
1	0	(0,,,)	0	(0,0,,)	0	(0,0,0,)	0	(0,0,0,0)
2	5	(1,,,)	5	(1,0,,)	5	(1,0,0,)	5	(1,0,0,0)
3	5	(1,,,)	7	(0,1,,)	7	(0,1,0,)	7	(0,1,0,0)
4	5	(1,,,)	7	(0,1,,)	7	(0,1,0,)	7	(0,1,0,0)
5	5	(1,,,)	12	(1,1,,)	12	(1,1,0,)	12	(1,1,0,0)
6	5	(1,,,)	12	(1,1,,)	12	(1,1,0,)	12	(1,1,0,0)
7	5	(1,,,)	12	(1,1,,)	12	(1,1,0,)	12	(1,1,0,0)
8	5	(1,,,)	12	(1,1,,)	13	(0,1,1,)	13	(0,1,1,0)
9	5	(1,,,)	12	(1,1,,)	13	(0,1,1,)	13	(0,1,1,0)

Таким образом, получаем оптимальное решение $(0, 1, 1, 0)$ и соответствующее значение целевой функции $g_4(9) = 13$. Трудоёмкость алгоритма равна $O(nC)$ операций, где C – размер ранца.

8.2.2 Графический подход

Функцию $g_\alpha(t)$ можно представить в следующем виде:

t	t_0	t_1	$\dots$	t_{m_α}
g	f_0	f_1	$\dots$	f_{m_α}

т.е. при $t \in [t_j, t_{j+1})$ имеем $g_\alpha(t) = f_j$, $j = 0, 1, \dots, m_\alpha - 1$.
Функцию $g_{\alpha+1}(t)$ можно получить из функции $g_\alpha(t)$ следующим образом:

$$g^1(t) = g_\alpha(t),\ x_{\alpha+1}(t) = 0,$$

$$g^2(t) = p_{\alpha+1} + g_\alpha(t - w_{\alpha+1}),\ x_{\alpha+1}(t) = 1,\ w_{\alpha+1} \le t,$$

$$g^2(t) = g^1(t),\ x_{\alpha+1}(t) = 0,\ w_{\alpha+1} > t,$$

$$g_{\alpha+1}(t) = \max\{g^1(t), g^2(t)\}.$$

График $g^2(t)$ строится из графика $g_\alpha(t)$ смещением “наверх” на $p_{\alpha+1}$ и “вправо” на $w_{\alpha+1}$. Функцию $g^2(t)$ можно представить в следующем виде

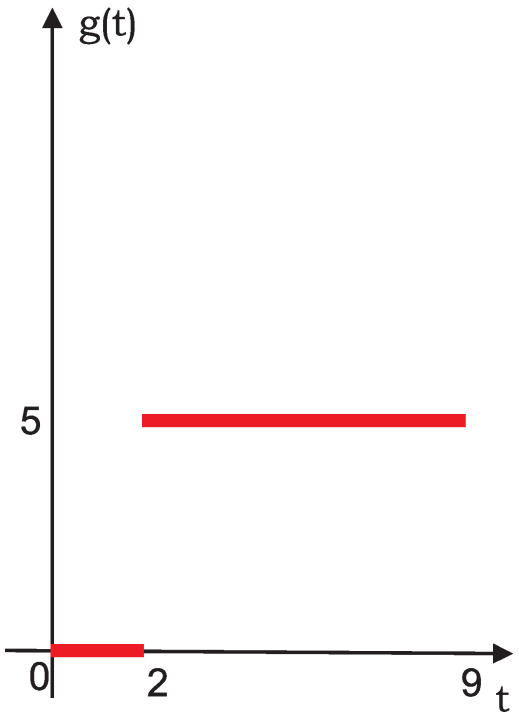

Рис. 8.5: Функция $g_1(t)$

t	$t_0 + w_{\alpha+1}$	$t_1 + w_{\alpha+1}$	$\ldots$	$t_{m_\alpha} + w_{\alpha+1}$
g	$f_0 + p_{\alpha+1}$	$f_1 + p_{\alpha+1}$	$\ldots$	$f_{m_\alpha} + p_{\alpha+1}$

График $g^1(t)$ полностью повторяет график $g_\alpha(t)$. Следовательно, чтобы построить $g_{\alpha+1}(t) = \max\{g^1(t), g^2(t)\}$, необходимо рассмотреть не более $2m_\alpha$ интервалов, образованных точками из множества $\{t_0, t_1, \ldots, t_{m_\alpha}, t_0 + a_{\alpha+1}, t_1 + a_{\alpha+1}, \ldots, t_{m_\alpha} + a_{\alpha+1}\}$, которые принадлежат интервалу $[0, C]$. Количество точек не превосходит C для $w_i \in Z^+$, $i = 1, \ldots, n$. Таким образом, трудоёмкость графического алгоритма для целочисленного примера не превосходит $\min\{O(nC), O(nf_{\max})\}$ операций, как и для алгоритма, основанного на методе динамического программирования. Необходимо также отметить, что параметры задачи могут быть как нецелочисленными, так и отрицательными. В этом случае график $g^2(t)$ строится из графика $g_\alpha(t)$ смещением "вниз" на $|p_{\alpha+1}|$ (когда $p_{\alpha+1} < 0$) и "влево" на $|w_{\alpha+1}|$ ($w_{\alpha+1} < 0$).

Покажем на том же примере (8.3) работу графического алгоритма.

Шаг 1. Результаты вычислений и целевая функция $g_1(t)$ представлены на рис. 8.5. В результате получаем следующие значения:

t	0	2
g	0	5
$x(t)$	$(0,,,)$	$(1,,,)$

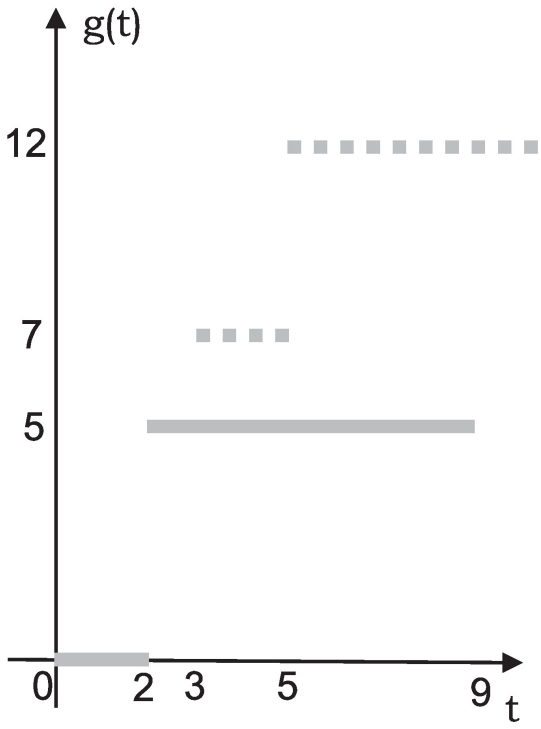

Рис. 8.6: Функции $g^1(t)$ и $g^2(t)$ (пунктиром)

Шаг 2. На рис. 8.6 представлены функции $g^1(t)$ и $g^2(t)$ (пунктиром). Для построения функции $g_2(t)$ необходимо рассмотреть интервалы, образованные точками $0, 2, 0+3, 2+3$. Результаты вычислений и целевая функция $g_2(t)$ представлены на рис. 8.7. В результате получим:

t	0	2	3	5
g	0	5	7	12
$x(t)$	$(0,0,,)$	$(1,0,,)$	$(0,1,,)$	$(1,1,,)$

Шаг 3. Для построения функции $g_3(t)$ необходимо рассмотреть интервалы, образованные точками $0, 2, 3, 5, 0+5, 2+5, 3+5$. Точка $5+5 > 9$ не рассматривается. Многие фрагменты $g^2(t)$ (обозначено пунктиром) "поглощаются" и не участвуют в $g_3(t)$. Результатом работы третьего шага алгоритма будет:

t	0	2	3	5	8
g	0	5	7	12	13
$x(t)$	$(0,0,0,)$	$(1,0,0,)$	$(0,1,0,)$	$(1,1,0,)$	(0,1,1,)

Шаг 4. Результаты вычислений и целевая функция $g_4(t)$ представлены на рис.8.8.

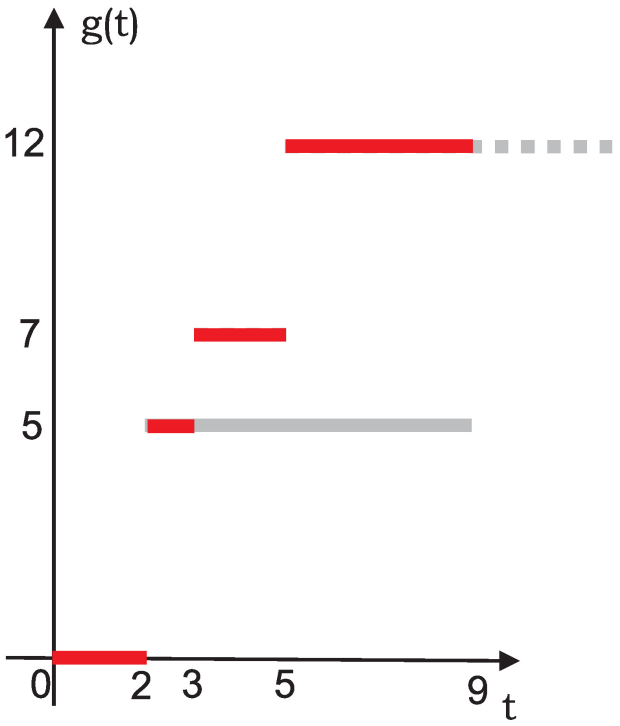

Рис. 8.7: Функция $g_2(t)$

Для построения функции $g_4(t)$ необходимо рассмотреть интервалы, образованные точками $0, 2, 3, 5, 8, 0+7, 2+7$. Точки $3+7, 5+7, 8+7$ не рассматриваются. Таким образом, достаточно рассмотреть 5 точек. В результате получим:

t	0	2	3	5	8
g	0	5	7	12	13
$x(t)$	$(0,0,0,0)$	$(1,0,0,0)$	$(0,1,0,0)$	$(1,1,0,0)$	(0,1,1,0)

8.2.3 Эффективность графического алгоритма

В процессе работы алгоритма было просмотрено всего 17 точек: на первом шаге 2 точки, на втором – 3 точки, на третьем – 7 точек, на четвертом шаге 5 точек. В алгоритме динамического программирования пришлось бы просмотреть $4 \cdot 9 = 36$ точек. Следовательно, для данного примера мы значительно сокращаем псевдополиномиальную составляющую трудоёмкости алгоритма.

Стоит отметить, что графический алгоритм работает и в случае, когда $w_i \notin Z,\ \forall i,\ C \notin Z$.

На шаге 3 и 4 видно, что удвоение количества “хранимых” интервалов не происходит. Стоит предположить, что для большого числа примеров

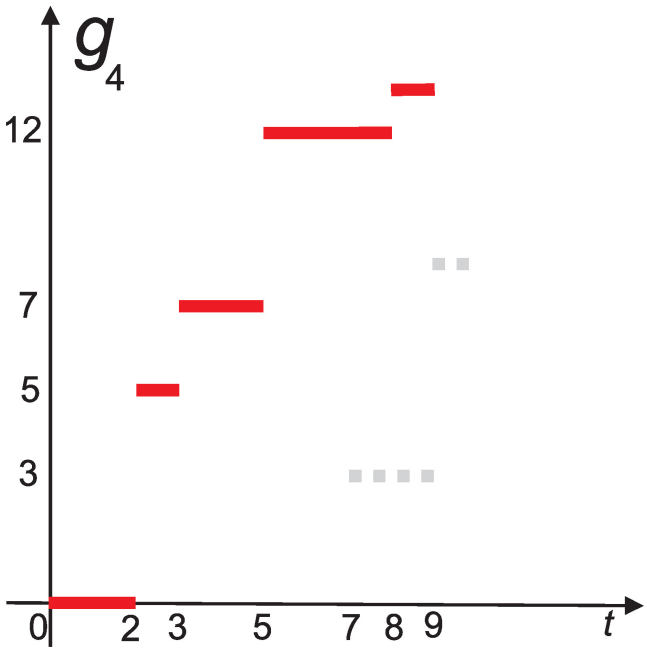

Рис. 8.8: Функция $g_4(t)$

трудоёмкость графического алгоритма будет полиномиальной.

Чтобы на шаге α появлялось как можно меньше новых "хранимых" интервалов необходимо упорядочить исходные данные в порядке $\frac{p_1}{w_1} \geq \frac{p_2}{w_2} \geq \geq \ldots \geq \frac{p_n}{w_n}$. Тогда функция $g^2(t)$ будет "поглощаться" эффективнее.

Алгоритм косвенно учитывает особенности задачи. В алгоритме динамического программирования никак не учитывается, что предмет 4 в приведенном примере наименее предпочтителен (см. $\frac{p_4}{w_4}$). В графическом алгоритме на шаге 4 заметно, что функция $g^2(t)$ не оказала влияния на $g_4(t)$, т.е. была учтена некоторая особенность задачи.

Идея графического подхода является естественным продолжением метода динамического программирования. Данный подход может быть применён к задачам как с нецелочисленными, так и с отрицательными значениями параметров.

Необходимо также отметить, что решение примера из [58], стр. 289:

$$\begin{cases} \sum\limits_{i:=1}^{n} p_i x_i \longrightarrow \max \\ \sum\limits_{i:=1}^{n} 2x_i \leq 2[\frac{n}{2}] + 1, \end{cases} \tag{8.4}$$

когда $p_i = 2$, $i = 1, \ldots, n$, при $n \geq 4$, может быть найдено с помощью графического подхода за $O(n)$ операций. В общем случае, при произвольных p_i, $i = 1, \ldots, n$, пример (8.4) будет решен за $O(n \log n)$ операций предлагаемым

алгоритмом. Алгоритмы, основанные на методе ветвей и границ находят решение данного примера за $O(\frac{2^n}{\sqrt{n}})$ операций.

В настоящее время проводится экспериментальное исследование предлагаемого подхода на тестовых примерах "большой" размерности, также будет проведен сравнительный анализ с алгоритмами, основанными на методе ветвей и границ (количество точек ветвления в дереве поиска с количеством "хранимых" интервалов).

Ниже приводится решение примера задачи методом динамического программирования и графическим алгоритмом [31].

$$\begin{cases} 5x_1 + 7x_2 + 6x_3 + 5x_4 + 4x_5 + 8x_6 + 6x_7 \longrightarrow \max \\ 2x_1 + 3x_2 + 5x_3 + 4x_4 + 2x_5 + 3x_6 + 4x_7 \leq 10 \\ x_j \in 0, 1, j = 1, \ldots, 7. \end{cases} \tag{8.5}$$

Для алгоритма динамического программирования необходимо вычислить таблицу с $10 \cdot 7 = 70$ элементов (см. таблицу 8.4).

Таблица 8.4:

C	1	2	3	4	5	6	7	8	9	10
1	0	5	5	5	5	5	5	5	5	5
2	0	5	7	7	12	12	12	12	12	12
3	0	5	7	7	12	12	12	13	13	18
4	0	5	7	7	12	12	13	13	17	18
5	0	5	7	9	12	12	16	16	17	18
6	0	5	8	9	13	15	17	20	20	24
7	0	5	8	9	13	15	17	20	20	24

Модифицированный алгоритм динамического программирования вычисляет 41 значение, см. таблицу 8.5

Таблица 8.5:

C	1	2	3	4	5	6	7	8	9	10
1	0	5								
2	0	5	7	7	12					
3	0	5	7	7	12	12	12	13	13	18
4	0	5	7	7	12	12	13	13	17	18
5			7	9	12	12	16	16	17	18
6						15	17	20	20	24
7										24

Таблица 8.6:

C	1	2	3	4	5	6	7	8	9	10
1	0	5								
2			7		12					
3								13		18
4							13		17	
5				9			16			
6						15	17	20		24
7										

Графический алгоритм вычисляет только 14 элементов, см. таблицу 8.6

Таким образом, на этом примере мы сократили вычисление на 80% по сравнению с методом динамического программирования.

Глава 9

Применение графического подхода

9.1 Одноприборные задачи

Задачи для одного прибора распространены в силу их тесной связи с классическими задачами из других областей дискретной математики, а также в связи с тем, что одноприборные задачи являются частными случаями и подзадачами более сложных практических задач. В этой главе представлены некоторые алгоритмы решения классических одноприборных задач.

Для одноприборных задач можно выделить следующее важное свойство.

Теорема 9.1 *Если* $r_j = 0$ *для всех* $j = 1, 2, \ldots, n$, *и целевая функция* $F(C_1, C_2, \ldots, C_n)$ *является монотонно неубывающей функцией, зависящей от моментов окончания обслуживания требований* C_j, *то в задаче минимизации функции* F *существует оптимальное расписание без прерываний обслуживания требований и простоев прибора.*

Доказательство. Предположим, что во всех оптимальных расписаниях происходят прерывания в обслуживании работ. Пусть π – оптимальное расписание, при котором обслуживание работы j прерывается. Пусть работа j обслуживается в интервалах времени $[t_1, t_2)$ и $[t_3, t_4)$, где $t_1 < t_2 < t_3 < t_4$, причем требование j не обслуживается в промежутке (t_2, t_3) и до момента времени t_1. Преобразуем расписание π следующим образом. Передвинем часть обслуживания $[t_1, t_2)$ вперед во времени, чтобы требование j обслуживалось без перерывов в промежутке $[t_3 - (t_2 - t_1), t_4)$. При этом требования, обслуживавшиеся в промежутке времени (t_2, t_3) "сдвинем назад" во времени на величину $t_2 - t_1$. В результате такой операции значение целевой функции не увеличилось, но при этом мы сократили одно прерывание. Повторив эту

операцию необходимое число раз, равное суммарному количеству всех прерываний, мы получим оптимальное расписание без прерываний. □

Вероятнее всего, ланное утверждение было ранее доказано более изящной техникой...

Многие известные нам целевые функции являются монотонно возрастающими (неубывающими) функциями, зависящими от моментов окончания обслуживания требований C_j. Например, $\sum C_j, \sum U_j, \sum T_j$ и т.д.

Приведенный факт можно использовать следующим образом. Если некоторая задача соответствует условиям теоремы 9.1, тогда оптимальное расписание для этой задачи однозначно задается перестановкой элементов множества N.

Определение 9.1 *Перестановка из n элементов – это конечная последовательность длины n, все элементы которой различны.*

Для данных задач перестановка вида $\pi = (j_1, j_2, \ldots, j_n)$, задающая расписание π, определяет порядок обслуживания требований (см. векторное представление расписаний). То есть первым обслуживается требование j_1, за ним – требование j_2 и т.д. Тогда моменты завершения обслуживания для каждого из требований $C_{j_k}(\pi) = \sum_{l=1}^{k} p_{j_l}$, $k = 1, 2, \ldots, n$. Величину $C_{j_k}(\pi)$ называют временем (моментом) завершения обслуживания требований j_k при расписании π.

Для задач, в которых расписание можно задать перестановкой, важными являются следующие определения:

Определение 9.2 *EDD (Earliest Due Date) порядок обслуживания требований – очередность обслуживания, при которой требования обслуживаются в порядке неубывания директивных сроков d_j.*

Определение 9.3 *LDD (Latest Due Date) порядок обслуживания требований – очередность обслуживания, при которой требования обслуживаются в порядке невозрастания директивных сроков d_j.*

Определение 9.4 *SPT (Shortest Processing Time) порядок обслуживания требований – очередность обслуживания, при которой требования обслуживаются в порядке неубывания времен обслуживания p_j.*

Определение 9.5 *LPT (Longest Processing Time) порядок обслуживания требований – очередность обслуживания, при которой требования обслуживаются в порядке невозрастания времен обслуживания* p_j.

Определение 9.6 *Частичное расписание* π' *– фрагмент целого расписания* π, *описывающее порядок обслуживания подмножества требований* $N' \subset N$.

В данной главе запись вида $\{\pi\}$ обозначает множество требований, обслуживаемых при расписании π. Запись вида $i \in \pi$ означает $i \in \{\pi\}$. Через $\pi \setminus \{i\}$, где $\pi = (\pi_1, i, \pi_2)$, будем обозначать частичное расписание вида (π_1, π_2). Запись $j \to i$ означает, что обслуживание требования j предшествует обслуживанию требования i. Соответственно, запись $(j \to i)_\pi$ означает, что это выполняется при расписании π.

9.2 Одноприборные задачи $1|p_j = p|\sum f_j$

В этом разделе предлагается алгоритм решения одноприборных задач, для которых $p_j = p,\ r_j \in Z^+,\ j = 1, 2, \ldots, n$, и нет отношений предшествования. Обозначим эти задачи как $1|r_j, p_j = p|\sum f_j$. При этом функция f_j требования j зависит от времени окончания обслуживания C_j. Данную задачу можно решить, сведя ее к ЗАДАЧЕ О НАЗНАЧЕНИЯХ.

Количество заданий в ЗАДАЧЕ О НАЗНАЧЕНИЯХ будет равно $r_{\max} + n$, а количество исполнителей равно n. То есть для каждого требования $j, j = 1, 2, \ldots, n$, нужно определить интервал обслуживания $[t, t+1) \in$ $\in [0, r_{\max} + n)$. Матрица стоимостей при этом формируется следующим образом. Для требования j для каждого $t \in [0, r_{\max} + n),\ j \geq r_j$, вычислим $a_{tj} = f_j(C_j)$, где $C_j = t + 1$. Для каждой точки $t < r_j$ и для каждой точки $t \geq D_j$ примем $a_{tj} = +\infty$.

Решив ЗАДАЧУ О НАЗНАЧЕНИЯХ, получим интервалы обслуживания каждого из требований. То есть задачу $1|p_j = p|\sum f_j$ можно решить за время $O(n^3)$ операций.

Рассмотрим следующую практическую задачу. У фермера есть один комбайн. Ему необходимо убрать урожай с n' полей. На уборку поля $j,\ j = 1, 2, \ldots, n'$, необходимо p_j дней. На полях в разное время засеяна разная культура, поэтому для каждого поля определен "идеальный день" d_j, к которому урожай на поле нужно убрать. Нарушение этого срока ведет к тому, что урожай портится. То есть возникают потери качества, пропорционально зависящие от количество дней запаздывания с уборкой и от ценности

культуры. Ценность культуры задается числом w_j. Обозначим через C_j время, к которому поле j убрано. Если $C_j - d_j = T_j > 0$, то потери качества для этого поля могут задаваться формулой $w_j(1 + 2 + 3 + \ldots + T_j)$. Необходимо минимизировать суммарные потери качества по всем культурам. Такую задачу можно решить следующим образом. Очевидно, что период сбора урожая, не превышает $5 * 31 = 150$ дней, т.е. $\sum p_j = 150$. Мы конструируем пример задачи $1|p_j = p|\sum f_j$, где $n = \sum_{j=1}^{n'} p_j$. Делим каждое поле j на p_j отдельных требований. Для каждого возможного дня t, для каждого требования k, $k = 1, 2, \ldots, n$, рассчитываем $a_{tk} = w_j \max\{0, C_k - d_j\}$, если требование k относится к полю j. Время решения этой задачи $O(n^3)$ будет приемлемым для любого современного компьютера, т.к. $n \leq 150$.

9.3 Минимизация числа запаздывающих требований $1||\sum U_j$

Имеется множество требований $N = \{1, 2, \ldots, n\}$. Для каждого требования $j \in N$ заданы продолжительность обслуживания $p_j > 0$ и директивный срок d_j, к которому требование должно быть обслужено. Прерывания при обслуживании и обслуживание более одного требования в любой момент времени запрещены. Расписание π однозначно задается перестановкой. Если $C_j(\pi) > d_j$, тогда требование j запаздывает, и в этом случае полагают $U_j = 1$. Если $C_j(\pi) \leq d_j$, тогда требование j не запаздывает, и $U_j = 0$. Необходимо построить расписание π, при котором значение целевой функции $F(\pi) = \sum_{j=1}^{n} U_j(\pi)$ минимально.

Далее представлен полиномиальный алгоритм решения данной задачи, который основан на следующей лемме.

Лемма 9.1 *Для каждого примера задачи $1||\sum U_j$ существует оптимальное расписание вида $\pi = (G, H)$, при котором все требования $i \in G$ не запаздывают, а все требования $j \in H$ запаздывают.*

Доказательство. Предположим, что существует оптимальное расписание вида $\pi = (j_1 \ldots, j_{k-1}, j_k, \ldots, j_n)$, при котором требование j_{k-1} запаздывает, а требование j_k не запаздывает. Тогда $\pi' = (j_1, \ldots, j_k, j_{k-1}, \ldots, j_n)$ также является оптимальным расписанием, т.к. $F(\pi') \leq F(\pi)$. После конечного числа попарных перестановок требований, каждая из которых не увеличивает значение целевой функции, получим оптимальное расписание вида $\pi = (G, H)$.

□

Докажем еще одно свойство, важное для задач, связанных с запаздыванием.

Лемма 9.2 *Если для примера задачи* $1||\sum U_j$ *существует оптимальное расписание, где все требования не запаздывают. То расписание* π_{EDD} *(требования обслуживаются в порядке EDD) также является оптимальным для данного примера.*

Доказательство. Предположим, что существует оптимальное расписание вида $\pi = (j_1 \ldots, j_{k-1}, j_k, \ldots, j_n)$, при котором требование $d_{j_{k-1}} > d_{j_k}$. Тогда расписание $\pi' = (j_1, \ldots, j_k, j_{k-1}, \ldots, j_n)$ также является оптимальным, т.к. $F(\pi') \leq F(\pi)$. Произведя данную попарную перестановку необходимое число раз, мы получим оптимальное расписание π_{EDD}. □

Очевидно, что данное свойство выполняется и для многих других задач таких, как $1||\sum w_j U_j$, $1||\sum w_j T_j$ и т.д. Теперь мы можем утверждать, что для задачи $1||\sum U_j$ существует оптимальное расписание, при котором все незапаздывающие требования обслуживаются в EDD порядке в начале расписания, а запаздывающие требования – в произвольном порядке в конце расписания.

Идея алгоритма решения данной задачи описывается следующим образом. Мы последовательно формируем множество незапаздывающих требований $\{G\}$. Требования включаются в множество $\{G\}$ в порядке неубывания директивных сроков и добавляются в конец частичного расписания G. Если после добавления очередного требования j окажется, что оно запаздывает, то мы находим требование $i \in G$, с максимальной продолжительностью обслуживания и "перемещаем" его в множество запаздывающих требований $\{H\}$. Формально алгоритм 9.1 решения задачи $1||\sum U_j$ описан ниже.

Алгоритм 9.1 был разработан Муром [176] в 1968 году и имеет трудоемкость $O(n \log n)$ операций.

Пример. Рассмотрим пример с 5-ю требованиями. Параметры требований: $p_1 = 2,\ p_2 = 5,\ p_3 = 3,\ p_4 = 2,\ p_5 = 4,\ d_1 = 3,\ d_2 = d_3 = 9,\ d_4 = d_5 = 10$. При исполнении алгоритма 9.1 включаем в множество G требования 1 и 2. При этом $t = 2 + 5 = 7 < 9 = d_2$. Пробуем включить требование 3. Получаем $t = 2 + 5 + 3 = 10 > 9 = d_3$, тогда находим в G требование с максимальной продолжительностью $i = 2$. Получаем $G = \{1, 3\},\ t = 2 + 3 = 5$. Включаем требование 4, имеем $t = 2 + 3 + 2 = 7 < 10 = d_4$. Добавляем в множество G требование 5. Имеем $t = 2 + 3 + 2 + 4 = 11 > 10 = d_5$, то находим в G

Алгоритм 9.1 Алгоритм решения задачи $1||\sum U_j$.

```
1: Перенумеруем требования согласно правилу d_1 ≤ d_2 ≤ ... ≤ d_n;
2: G := ∅; H := ∅; t := 0;
3: for j := 1 to n do
4:    G := G ⋃ {j};
5:    t := t + p_j;
6:    if t ≥ d_j then
7:       Найдем требование i ∈ G с максимальной продолжительность обслуживания, т.е.
         i := argmin_{k∈G} p_k;
8:       G := G \ {i};
9:       H := H ⋃ {i};
10:      t := t − p_i;
11:   end if
12: end for
```

требование с максимальной продолжительностью $i = 5$ и исключаем его из G. Имеем в итоге $G = \{1, 3, 4\}$ и оптимальное расписание $\pi = (1, 3, 4, 2, 5)$.

Напоминаем, что для задачи $1||\sum U_j$ рассматриваются только расписания вида $\pi = (G, H)$.

Лемма 9.3 *Пусть при расписании* $\pi = (j_1, j_2, \ldots, j_l, j_{l+1}, \ldots, j_n)$ *выполняется* $d_{j_1} \le d_{j_2} \le \ldots \le d_{j_n}$. *Требование* j_l *– последнее незапаздывающее требование. Тогда существует оптимальное расписание, при котором требование* $j^* \in \{j_1, j_2, \ldots, j_{l+1}\}$, $p_{j^*} \ge p_i$, *для всех* $i \in \{j_1, j_2, \ldots, j_{l+1}\}$, *запаздывает.*

Доказательство. Предположим, что существует оптимальное расписание $\pi = (\pi_1, j^*, \pi_2, \pi_3, i, \pi_4)$, при котором требования множества $\{\pi_1, j^*, \pi_2, \pi_3\}$ не запаздывают, а требования $\{i, \pi_4\}$ запаздывают. Согласно лемме 9.2 для рассматриваемого примера не существует расписания, при котором все требования $j_1, j_2, \ldots, j_l, j_{l+1}$ не запаздывают. Тогда существует запаздывающее требование $i \in \{j_1, j_2, \ldots, j_l, j_{l+1}\}$, $i \ne j^*$. В соответствии с леммой 9.2 все требования $\{\pi_1, j^*, \pi_2, \pi_3\}$ обслуживаются в порядке EDD.

Если $d_{j^*} \le d_i$, то расписание $\pi' = (\pi_1, i, \pi_2, j^*, \pi_3, \pi_4)$ также оптимальное, т.к. $F(\pi') \le F(\pi)$. При расписании π' требование j^* запаздывает, а требование i не запаздывает.

Рассмотрим случай $d_{j^*} > d_i$. Пусть для всех $j \in \{\pi_3\}$, $d_j \ge d_i$, и для всех $j \in \{\pi_2\}$ выполняется $d_j < d_i$.

Рассмотрим расписание $\pi'' = (\pi_1, \pi_2, i, \pi_3, j^*, \pi_4)$. Для всех $j \in \{\pi_2\} \bigcup \{\pi_3\}$ имеем $C_j(\pi'') \ge C_j(\pi)$, так как $p_{j^*} \ge p_i$. Пусть k – последнее требование из

частничного расписания π_2. Тогда $C_i(\pi'') \leq C_k(\pi') \leq d_k < d_i$, т.е. при расписании π'' все требования из множества $\{\pi_2\} \bigcup \{i\} \bigcup \{\pi_3\}$ не запаздывают. Поэтому расписание π'' оптимальное. Лемма доказана. $\square$

Теорема 9.2 *При помощи алгоритма 9.1 для задачи* $1||\sum U_j$ *за* $O(n \log n)$ *операций строится оптимальное расписание.*

Доказательство. Докажем теорему методом математической индукции. Алгоритм верен при $n = 1$.

Предположим, что алгоритм верен для всех примеров размерности $n-1$ требований. Рассмотрим пример размерности n требований. Предположим, что при помощи алгоритма 9.1 было построено расписание $\pi = (G, H)$, но существует оптимальное расписание $\pi' = (G', H')$ такое, что $|G'| \geq |G|$.

При использовании алгоритма 9.1 для $n-1$ требований $1, 2, \ldots, j-1$, $j+1, \ldots, n$, где требование $j = j^*$ выбрано согласно лемме 9.3, будет получено оптимальное расписание $(G \setminus \{j^*\}, H)$. Расписание $(G' \setminus \{j^*\}, H')$ является допустимым для соответствующего примера из $n-1$ требований. Поэтому имеем $|G'| \leq |G|$, тогда $|G'| = |G|$. Значит алгоритм строит оптимальное решение.

Трудоемкость сортировки требований в порядке EDD на первом шаге алгоритма равна $O(n \log n)$ операций. Трудоемкость цикла 3–12 равна $O(n)$ операций. То есть трудоемкость алгоритма равна $O(n \log n)$ операций. $\square$

9.4 Минимизация взвешенного числа запаздывающих требований $1||\sum w_j U_j$

Очевидным является тот факт, что задача минимизации взвешенного числа запаздывающих требований эквивалентна задаче максимизации взвешенного числа **не**запаздывающих требований.

То есть целевая функция $F(\pi) = \sum w_j U_j(\pi) \to \min$ может быть заменена на целевую функцию $F(\pi) = \sum w_j [1 - U_j(\pi)] \to \max$.

Теорема 9.3 *Задача* $1||\sum w_j U_j$ *является* **NP-трудной**.

Доказательство. Доказательство проведем сведением ЗАДАЧИ О РАНЦЕ к частному случаю задачи $1||\sum w_j U_j$ с общим директивным сроком $d_j = d$. Множеству предметов $N = \{1, 2, \ldots, n\}$ ЗАДАЧИ О РАНЦЕ ставим

в соответствие множество требований $N' = \{1, 2, \ldots, n\}$ задачи $1||\sum w_j U_j$. Пусть вес требования $w_j = p_j$, продолжительность обслуживания требования $p_j = w_j$ для каждого j, $j = 1, 2, \ldots, n$. Зададим общий директивный срок $d_j = C$, $j = 1, 2, \ldots, n$. Тогда при оптимальном расписании π^* значение $\sum w_j(1 - U_j(\pi^*))$ равно максимальной суммарной стоимости предметов, помещенных в рюкзак, то есть равно значению целевой функции для ЗАДАЧИ О РАНЦЕ. Таким образом, сводимость доказана.

Очевидно, что сведение полиномиально, т.е. выполняется за полиномиальное число шагов. □

Теорема 9.4 *Для задачи $1||\sum w_j U_j$ существует оптимальное расписание вида $\pi = (G, H) = (EDD, LDD)$, где все требования $j \in H$ запаздывают, а все требования $i \in G$ не запаздывают. Требования из множества G обслуживаются в порядке EDD, а требования из множества H – в порядке LDD.*

Доказательство этой теоремы аналогично доказательству лемм 9.1 и 9.2. Заметим, что в оптимальном расписании незапаздывающие требования обслуживаются в порядке EDD, в то время как запаздывающие требования могут обслуживаться в любом порядке. Алгоритм решения 9.2 основан на теореме 9.4.

В данном алгоритме $\pi_j(t)$ означает оптимальное частичное расписание для требований $1, 2, \ldots, j$, при котором обслуживание требований начинается в момент времени t, а $f_j(t) = \sum_{i=1}^{j} w_i[1 - U_i(\pi_j(t))]$ соответствует максимальному взвешенному числу незапаздывающих требований при этом частичном расписании.

Теорема 9.5 *Алгоритм 9.2 строит оптимальное расписание задачи максимизации взвешенного числа незапаздывающих требований за $O(n\sum p_j)$ операций.*

Доказательство. Доказательство от противного.
Допустим, что существует оптимальное расписание $\pi^* = (EDD, LDD)$, для которого $f(\pi^*) > f(\pi_n(0)) = f_n(0)$.

Пусть $\pi' := \pi^*$. Для каждого $l = 1, 2, \ldots, n$, мы последовательно рассматриваем часть $\bar{\pi}_l \in \pi'$, $\{\bar{\pi}_l\} = \{1, \ldots, l\}$, расписания. Пусть $\pi' = (\pi_\alpha, \bar{\pi}_l, \pi_\beta)$. Если $\bar{\pi}_l \neq \pi_l(t)$, где $t = \sum_{i \in \pi_\alpha} p_i$, тогда примем $\pi' := (\pi_\alpha, \pi_l(t), \pi_\beta)$. Очевидно,

Алгоритм 9.2 Алгоритм решения задачи $1||\sum w_j U_j$.

Перенумеруем требования согласно правилу $d_1 \leq d_2 \leq \ldots \leq d_n$;
for $t := 0$ to $\sum_{i=2}^{n} p_i$ **do**
  $\pi_1(t) := (1)$;
  if $p_1 + t - d_1 \leq 0$ **then**
    $f_1(t) := w_1$;
  else
    $f_1(t) := 0$;
  end if
end for
for $j := 2$ to n **do**
  for $t := 0$ to $\sum_{i=j+1}^{n} p_i$ **do**
    $\pi^1 := (j, \pi_{j-1}(t + p_j))$, $\pi^2 := (\pi_{j-1}(t), j)$;
    if $p_j + t - d_j \leq 0$ **then**
      $\Phi^1(t) := w_j + f_{j-1}(t + p_j)$;
    else
      $\Phi^1(t) := f_{j-1}(t + p_j)$;
    end if
    if $\sum_{i=1}^{j} p_i + t - d_j \leq 0$ **then**
      $\Phi^2(t) := f_{j-1}(t) + w_j$;
    else
      $\Phi^2(t) := f_{j-1}(t)$;
    end if
    if $\Phi^1(t) < \Phi^2(t)$ **then**
      $f_j(t) := \Phi^2(t)$;
      $\pi_j(t) := \pi^2$;
    else
      $f_j(t) := \Phi^1(t)$;
      $\pi_j(t) := \pi^1$;
    end if
  end for
end for

$\pi_n(0)$ – оптимальное расписание, а $f_n(0)$ – соответствущее оптмальное значение целевой функции (максимизации взвешенного числа незапаздывающих требований).

Таблица 9.1: Функция $f_j(t)$

t	t_1	t_2	$\ldots$	t_{m_j}
$f_j(t)$	W_1	W_2	$\ldots$	W_{m_j}
оптим. частичное расписание	π_1	π_2	$\ldots$	π_{m_j}

что $f((\pi_\alpha, \bar{\pi}_l, \pi_\beta)) \leq f((\pi_\alpha, \pi_l(t), \pi_\beta))$. Аналогичную операцию проделываем для каждого $l = 1, 2, \ldots, n$. В конце получаем $f(\pi^*) \leq f(\pi') \leq f_n(0)$. Поэтому расписание $\pi_n(0)$ так же оптимально.

Очевидно, что трудоемкость алгоритма составляет $O(n \sum p_j)$ операций, так как в цикле 10–31 выполняется $n-1$ итераций и просматриваются все целочисленные точки, ограниченного интервалом $\left[0, \sum_{j=1}^{n} p_j\right]$.

□

Фактически, в представленном алгоритме 9.2 динамического программирования используются следующие функциональные уравнения Беллмана:

$$f_j(t) = \max \begin{cases} \Phi^1(t) = \alpha(t) + f_{j-1}(t + p_j), j = 1, 2, \ldots, n; \\ \Phi^2(t) = \beta(t) + f_{j-1}(t), \qquad j = 1, 2, \ldots, n. \end{cases} \tag{9.1}$$

9.5 Графический алгоритм для задачи $1||\sum w_j U_j$

Идея графического алгоритма (GrA) для данной задачи можно описать следующим образом. На каждом шаге j алгоритма GrA мы сохраняем функцию $f_j(t)$ в табличном виде, представленном в таб. 9.1, где $t_1 < t_2 < \ldots < t_{m_j}$ и $W_1 > W_2 > \ldots > W_{m_j}$.

Записи в таблице означают следующее. Для каждого значения $t \in (t_l, t_{l+1}]$, $1 \leq l < m_j$, оптимальным будет расписание $\pi_l = (G, H) = (EDD, LDD)$, которому соответствует значение $f_j(t) = W_l = \sum_{i \in G} w_i$ (суммарное взвешенное число незапаздывающих требований). Точки t_l называются *точками излома*, т.е. выполняется $f_j(t') > f_j(t'')$ для $t' \leq t_l < t''$.

На очередном шаге $j+1$ мы трансформируем функцию $f_j(t)$ в две функции $\Phi^1(t)$ и $\Phi^2(t)$ согласно шагам 12–28 алгоритма 9.2 за $O(m_j)$ операций. Функция $\Phi^1(t)$ получается смещением графика функции $f_j(t)$ влево на p_{j+1} и добавлением новой точки $t = p_{j+1} - d_{j+1}$, а функция $\Phi^2(t)$ – добавлени-

Таблица 9.2: Сокращение числа интервалов для функции $f_{j+1}(t)$

t	$\ldots$	t_l	t_{l+1}	$\ldots$
$f_{j+1}(t)$	$\ldots$	W_l	$W_{l+1} = W_l$	$\ldots$
оптим. частичное расписание	$\ldots$	π_l	π_{l+1}	$\ldots$

ем новой точки $t = \sum_{i=1}^{j+1} p_i - d_{j+1}$. Все эти операции производятся при помощи таблиц вида 9.1. В каждой таблице $\Phi^1(t)$ и $\Phi^2(t)$ будет не более $m_j + 1$ точек излома. Далее мы вычисляем таблицу для функции

$$f_{j+1}(t) = \max\{\Phi^1(t), \Phi^2(t)\}$$

за $O(m_j)$ операций. В новой таблице, соответствующей функции $f_{j+1}(t)$, будет не более $2m_j + 2$ точек излома (обычно этом число меньше). Фактически, мы рассматриваем не все точки t из интервала $[0, \min\{d_j - p_j, \sum_{i=j+1}^{n} p_i\}]$, но только точки интвервала в которых меняется суммарное взвешенное число незапаздывающих требований.

Если встречается ситуация, описанная в таб. 9.2, то мы удаляем столбец, соответствующий точке t_{l+1} и принимаем $\pi_l := \pi_{l+1}$.

Очевидно, что на каждом шаге *GrA* будет рассмотрено не более F_{opt} точек излома, где $F_{opt} \leq \sum_{j=1}^{n} w_j$ – максимальное взвешенное число незапаздывающих требований. На каждом шаге $j = 1, 2, \ldots, n$ алгоритма *GrA* необходимо рассмотреть не более $\min\{2^j,\ d_j - p_j,\ \sum_{i=j+1}^{n} p_i, \sum_{i=1}^{j} w_i, F_{opt}\}$ точек излома, т.е. трудоемкость *GrA* не превосходит $O(\min\{2^n, n \cdot \min\{d_{\max}, F_{opt}\}\})$ операций.

9.6 Минимизация суммарного запаздывания $1||\sum T_j$

В данном параграфе рассматривается задача минимизации суммарного запаздывания $\sum T_j$, где $T_j = \max\{0, C_j - d_j\}$. То есть необходимо найти расписание π^*, при котором функция $F(\pi) = \sum_{j=1}^{n} T_j(\pi)$ достигает своего минимума. Доказательство ее NP-**трудности** было получено в 1990-м году, выяснение ее трудоемкости было нетривиальной проблемой. Далее для этой задачи будут представлены точный и метаэвристический алгоритмы решения, а также приближенный алгоритм с регулируемой точностью.

9.6.1 Точный алгоритм решения задачи $1||\sum T_j$

Алгоритм основан на следующих результатах, опубликованных Е.Лаулером в 1977-м году. Необходимо отметить, что данный алгоритм строит точное решение и для частного случая задачи $1||\sum w_j T_j$, при котором выполняется правило "если $p_j > p_i$, то $w_j \leq w_i,\ i, j \in N$".

Теорема 9.6 [150] *Пусть π – оптимальное расписание для примера задачи с директивными сроками $d_1, d_2, \ldots, d_n$ и пусть C_j – моменты завершения обслуживания требований $j = 1, 2, \ldots, n$ при этом расписании. Выберем новые директивные сроки d'_j так, что:*

$$\min\{d_j, C_j\} \leq d'_j \leq \max\{d_j, C_j\}.$$

Тогда любое оптимальное расписание π', соответствующее примеру с директивными сроками $d'_1, d'_2, \ldots, d'_n$ является оптимальным и для примера с исходными директивными сроками $d_1, d_2, \ldots, d_n$.

Доказательство. Обозначим через T суммарное запаздывания для примера с директивными сроками $d_1, d_2, \ldots, d_n$, а через T' – суммарное запаздывания для примера с директивными сроками $d'_1, d'_2, \ldots, d'_n$. Пусть π' – оптимальное расписание для примера с директивными сроками $d'_1, d'_2, \ldots, d'_n$, а C'_j – момент завершения обслуживания требования j при данном расписании. Запишем значения $T(\pi)$ и $T(\pi')$ в следующей форме:

$$T(\pi) = T'(\pi) + \sum_{j=1}^{n} A_j,$$

$$T(\pi') = T'(\pi') + \sum_{j=1}^{n} B_j,$$

где A_j и B_j – значения, зависящие от требования $j \in N$, которые вычисляются следующим образом:

(a) Пусть $C_j \leq d_j$, тогда выполняется:

1. Если $C'_j \leq d'_j$, тогда $A_j = 0$ и $B_j = 0$;
2. Если $d'_j < C'_j \leq d_j$, тогда $A_j = 0$ и $B_j < 0$;
3. Если $d_j < C'_j$, тогда $T_j(\pi') \leq T'_j(\pi')$. Поэтому $A_j = 0$ и $B_j < 0$.

(b) Пусть $C_j > d_j$, тогда выполняется:

1. Если $C'_j \le d_j$, тогда $A_j = d'_j - d_j \ge 0$ и $B_j = 0$;

2. Если $d_j < C'_j \le d'_j$, тогда $A_j = d'_j - d_j \ge 0$ и $B_j = (C'_j - d_j) - 0 < d'_j - d_j = A_j$;

3. Если $d'_j < C'_j$, тогда $A_j = B_j = d'_j - d_j$.

Выполняется $A_j \ge B_j$ для всех $j = 1, 2, \ldots, n$.
Более того, $T'(\pi) \ge T'(\pi')$, так как π' оптимальное расписание. Поэтому, получаем

$$T'(\pi) + \sum_{j=1}^{n} A_j \ge T'(\pi') + \sum_{j=1}^{n} B_j,$$

т.е. $T(\pi) \ge T(\pi')$. Теорема доказана. □

Лемма 9.4 *Для задачи* $1||\sum T_j$ *существует оптимальное расписание* π, *при котором:*

- *обслуживание требования* i *предшествует обслуживанию требования* j, *если* $d_i \le d_j$ *и* $p_i < p_j$;
- *все незапаздывающие требования обслуживаются в порядке неубывания директивных сроков.*

Доказательство. Сначала докажем первое свойство оптимальных расписаний. Пусть для двух требований i и j выполняется $d_i \le d_j$ и $p_i < p_j$. Предположим, что существует оптимальное расписание $\pi = (\pi_1, j, \pi_2, i, \pi_3)$. Для расписания $\pi' = (\pi_1, i, \pi_2, j, \pi_3)$ выполняется $F(\pi) - F(\pi') \ge (T_j(\pi) - T_j(\pi')) + (T_i(\pi) - T_i(\pi'))$.
Если $C_i(\pi') > d_j$, тогда $F(\pi) - F(\pi') \ge -(p_i + \sum_{k\in\pi_2} p_k) + (p_i + \sum_{k\in\pi_2} p_k) > 0$.
Иначе, если $C_i(\pi') \le d_j$, выполняется $F(\pi) - F(\pi') \ge -\max\{0, C_j(\pi') - d_j\} + \max\{0, C_j(\pi') - d_i\} \ge 0$, где $C_j(\pi') = C_i(\pi)$. Таким образом, первое свойство леммы доказано.

Докажем второе свойство. Пусть существует оптимальное расписание $\pi = (\pi_1, j, \pi_2, i, \pi_3)$, при котором $d_j > d_i$ и оба требования i и j не запаздывают. Тогда расписание $\pi' = (\pi_1, \pi_2, i, j, \pi_3)$ также является оптимальным. Повторяя подобную операцию необходимое количество раз получим оптимальное расписание, удовлетворяющее второму свойству леммы. □

Теорема 9.7 *Пусть требования упорядочены согласно правилу* $d_1 \le d_2 \le \ldots \le d_n$. *Обозначим через* j^* *– требование с максимальной продолжительностью обслуживания, т.е.* $j^* = \arg\max\limits_{j \in N}\{d_j : p_j = \max\limits_{i \in N} p_i\}$. *Тогда существует требование* $k \ge j^*$ *такое, что при оптимальном расписании все требования* $i, i = 1, 2, \ldots, k,\ i \ne j^*$, *обслуживаются перед требованием* j^*, *а остальные требования после требования* j^*.

Доказательство. Пусть C' – наибольший момент окончания обслуживания требования j^* при всех оптимальных расписаниях, соответствующих примеру с директивными сроками $d_1 \le d_2 \le \ldots \le d_n$. Обозначим через π оптимальное расписание, соответствующее примеру с модифицированными директивными сроками:

$$d_1, d_2, \ldots d_{j^*-1}, d'_{j^*} = \max\{d_{j^*}, C'\}, d_{j^*-1},\ \ldots, d_n,$$

и которое соответствует условиям леммы 9.4. Расписание π является оптимальным и для исходного примера согласно теореме 9.6. Пусть C – время завершения обслуживания требования j^* при расписании π. Тогда имеем $C \le C' \le d'_{j^*} = \max\{d_{j^*}, C'\}$, т.е. требование j^* не запаздывает. При расписании π все требования $i,\ d_i > d'_{j^*}$ обслуживаются перед требованием j^*, т.к. иначе требование i не будет запаздывающим, что противоречит второму свойству леммы 9.4. С другой стороны, все требования j такие, что $d_j \le d'_{j^*}$, должны обслуживаться при этом расписании перед требованием j^*, так как $p_j < p_{j^*}$ (см. лемму 9.4). □

Можно сделать следующий вывод из данной теоремы. Если выбрать такое требование j^* и для него определить требование (позицию) $k \ge j^*$, то исходную задачу можно разбить на две подзадачи решаемые аналогично. Первая подзадача содержит множество требований $i, i = 1, 2, \ldots, k,\ i \ne j^*$, а другая – множество требований $k+1, k+2, \ldots, n$. Сложность заключается в выборе требования (позиции) k.

На этом факте основан точный алгоритм 9.3 решения задачи.

Обозначим через $j^*(N')$ требование с наибольшей продолжительностью обслуживания среди требований множества $N' \subseteq N$. Если таких требований несколько, то выбирается требование с наибольшим директивным сроком, т.е. $j^*(N') = \arg\max\limits_{j \in N'}\{d_j : p_j = \max\limits_{i \in N'} p_i\}$. Для сокращения записи вместо $j^*(N')$ будем записывать j^*, если очевидно о каком множестве идет речь.

Рассмотрим пример (подпример) обслуживания требований множества $N' \subseteq N$, $N' = \{1, 2, \ldots, n'\}$, с момента времени $t' \ge 0$. Множество $L(N', t')$ есть множество всех индексов $i \in \{j^*, j^*+1, \ldots, n'\}$.

Алгоритм 9.3 Алгоритм решения задачи $1||\sum T_j$.

Procedure ProcL (N, t)

1: Дан пример $\{N, t\}$ с множеством требований $N = \{j_1, j_2, \ldots, j_n\}$ и моментом начала обслуживания t, где $d_{j_1} \leq d_{j_2} \leq \ldots \leq d_{j_n}$.
2: **if** $N = \emptyset$ **then**
3: π^*:= пустое расписание;
4: **else**
5: Найдем требование j^* из множества N;
6: Найдем множество $L(N, t)$ для требования j^*;
7: **for** ALL $k \in L(N, t)$ **do**
8: $\pi_k := (\mathbf{ProcL}(N', t'), j^*(N), \mathbf{ProcL}(N'', t''))$, где

$$N' := \{j_1, \ldots, j_k\} \setminus \{j^*\},\ t' := t,$$

$$N'' := \{j_{k+1}, \ldots, j_n\},\ t'' := t + \textstyle\sum_{i:=1}^{k} p_{j_i};$$

9: **end for**
10: π^*:=arg $\min\limits_{k \in L(N,t)} \{F(\pi_k, t)\}$;
11: **end if**
12: **RETURN** π^*;

Алгоритм решения.

1: π^*:=**ProcL**$(N, 0)$;

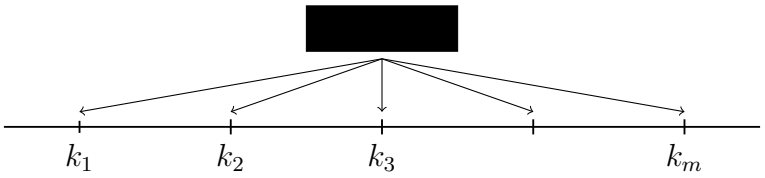

Рис. 9.1: Процесс выбора позиции k

Запись $F(\pi_k, t)$ в алгоритме означает суммарное запаздывание при расписании π_k выполнение которого начинается с момента времени t. На следующих двух рисунках показан процесс выбора позиции k и разбиение исходной задачи на две подзадачи.

Список рассматриваемых позиций $L(N', t')$ можно значительно сократить, т.к. в это множество стоит включать только требования $i \in \{j^*, j^* + + 1, \ldots, n'\}$, для которых выполняются условия:

Рис. 9.2: Разбиение исходной задачи на две подзадачи

(а) $t' + \sum_{j=1}^{k} p_j < d_{k+1}$ (правило исключения 1) и

(б) $d_j + p_j \le t' + \sum_{j=1}^{k} p_j$, для всех $j = \overline{j^*(N')+1, k}$
(правила исключения 2, 3),

где $d_{n'+1} := +\infty$.

Существуют и другие правила исключения, позволяющие сократить время работы алгоритма.

Теорема 9.8 [150] *Трудоемкость алгоритма 9.3 составляет* $O(n^4 \sum p_j)$ *операций.*

Модификации данного алгоритма с различными правилами сокращения перебора периодически появляются в литературе. Но на текущий момент, лучший из предложенных алгоритмов может решать лишь "простые" примеры, построенные по схеме Поттса и Вассенхова [186], размерностью не больше 500 требований.

9.7 Эффективность алгоритмов для тестовых примеров Поттса и Ван Вассенхова

Качество разных алгоритмов решения некоторой задачи комбинаторной оптимизации зачастую сравнивают экспериментально. При этом можно сравнить реальное время работы алгоритмов, количество итераций, производимых алгоритмом, точность полученного решения и т.д. В качестве иллюстрации того, как могут проводится эксперименты, в данном параграфе представлены результаты экспериментального сравнения Гибридного алгоритма и алгоритма ACO. Эксперименты проводились для примеров размерности $n = 4, \ldots, 70, 100$, которые генерировались с помощью схемы Поттса и Ван Вассенхова [186].

Примеры генерировались следующим образом. Значения $p_j \in Z$ распределены по равномерному закону на промежутке $[1, 100]$. Значения d_j генерировались по равномерному закону на промежутке

$$[\sum_{j:=1}^{n} p_j(1 - TF - RDD/2), \sum_{j:=1}^{n} p_j(1 - TF + RDD/2)].$$

Параметры TF и RDD принимают значения из множества $\{0.2, 0.4, 0.6, 0.8, 1\}$. Для каждой комбинации (TF, RDD) генерировалось по 100 примеров (всего 2500 примеров для каждого n).

Примеры, для которых $F(\pi_{EDD}) = 0$, не рассматривались, т.к. в этом случае Гибридный алгоритм и ACO гарантированно находят точные решения. π_{EDD}–расписание построенное по правилу EDD.

В алгоритмах использовались следующие параметры: $\alpha = 1$; $\beta = 2$; $\rho = 0{,}1$. Локальный поиск – попарная перестановка. Применяемая эвристика – MDD.

Для каждого примера запускался точный алгоритм 9.3. В результате его работы мы получали оптимальное значение целевой функции F_{opt}.

В алгоритме ACO муравьи генерировались до тех пор, пока не находилось расписание π, при котором $F(\pi) = F_{opt}$. Это "необходимое" количество муравьев фиксировалось. Число муравьев нами было ограничено $m \leq 100$. Перед первой итерацией параметры τ_{ij} рассчитывались по формуле $\tau_{ij} = 1/(T_{EDD}),\ i = 1, \ldots, n, j = 1, \ldots, n,\ \tau_0 = 1/(T_{EDD})$.

Алгоритм запускался до 10 раз (пока не найдено оптимальное расписание). Таким образом мы пытались избежать "случайности" работы алгоритма.

Лучшее найденное значение целевой функции F_{ACO} фиксировалось. Вычислялась относительная погрешность $\frac{F_{ACO} - F_{opt}}{F_{opt}}$.

Такая же схема использовалась для Гибридного алгоритма. Таким образом мы могли сравнить относительную погрешность двух алгоритмов (ACO и Гибридного), количество муравьев, необходимое каждому алгоритму для нахождения оптимального расписания.

Результаты экспериментов представлены в таб. 9.3.

Для размерности задачи $n = 100$ рассмотрено 617 примеров. Для остальных размерностей по 2500 примеров. В таблице выводятся только те строки, соответствующие размерности задачи n, для которых число неточно решенных примеров больше 0.

В первой колонке представлено значение n – размерность задачи. Во второй и третьей колонках, соответственно, число примеров, для которых алгоритм ACO или Гибридный не нашли точные решения. В четвертой и пятой колонках представлена максимальная относительная погрешность алгоритмов (с точностью до сотых процента). В последних двух колонках – среднее число муравьев, необходимое для поиска оптимального решения.

Таблица 9.3: Результаты экспериментов

n	Число неоптимально решенных примеров		Максимальная относительная погрешность		Число муравьев	
	ACO	Гибридный	*ACO*	Гибридный	*ACO*	Гибридный
19	2	0	0,22	0	1,6164	1,4004
20	1	0	0,58	0	1,6064	1,4204
22	1	0	0,16	0	1,626	1,4844
28	2	0	0,09	0	1,9704	1,6688
34	1	0	0,15	0	2,2212	1,9568
36	0	1	0	0,04	2,2332	2,154
37	1	1	0,38	0,01	2,4796	2,102
40	1	0	0,04	0	2,6036	2,2424
42	1	1	0,05	0,01	2,7888	2,4092
43	1	1	0,07	0,06	2,7316	2,3656
44	3	0	0,04	0	2,8464	2,3784
45	2	0	0,68	0	2,9736	2,4728
46	1	0	0,03	0	3,1624	2,4088
47	2	0	0,01	0	3,248	2,5152
48	9	0	0,56	0	3,4516	2,5196
49	3	1	0,15	0,08	3,4252	2,7
50	9	1	0,35	0,29	3,716	2,6336
51	8	0	0,22	0	3,8412	2,7768
52	4	1	0,04	0,07	3,5816	2,86
53	4	2	0,03	0,42	3,8948	2,9668
54	9	3	0,1	0,29	4,0324	2,9924
55	8	2	0,11	0,06	4,1048	3,0496
56	9	1	0,83	0,01	4,2916	3,0064
57	7	0	0,23	0	4,1568	3,158
58	14	0	0,17	0	4,71	3,3724
59	14	4	0,24	0,1	4,81	3,3372
60	11	1	0,22	0,01	4,7268	3,4224
61	18	2	1,26	0,02	5,3032	3,5216
62	10	2	0,26	0,01	5,0964	3,5032
63	17	7	0,16	0,08	5,3016	3,5728
64	15	6	0,57	0,46	5,2388	3,6504
65	18	7	0,1	0,14	5,548	3,6604
66	17	11	0,15	0,14	5,4288	3,8552
67	17	7	0,83	0,1	6,1068	4,1016
68	25	4	0,2	0,08	6,3864	3,7252
69	18	6	0,12	0,1	6,1912	4,0796
70	33	4	0,23	0,05	6,974	3,8672
100	36	0	0,31	0	27,35	4,66

Как видно из таблицы, примерно для 99% примеров оба алгоритма находят точные решения.

Число примеров, неточно решенных Гибридным алгоритмом, значительно меньше, чем алгоритмом ACO, и не превосходит 0,44% от общего числа примеров. Относительная погрешность не превосходит 0,46%. Число муравьев, необходимое Гибридному алгоритму также меньше. Заметно, что для $n = 100$ Гибридному алгоритму в среднем необходимо 4-5 муравьев для нахождения оптимального расписания, в то время как Алгоритму ACO требуется в среднем более 27 муравьев.

Относительная погрешность Алгоритма ACO превосходит 1,26% для $n = 61$. Число "неточно" решенных примеров для $n = 70$ больше 1% от общего числа примеров. Следует ожидать, что с ростом n будет наблюдаться значительное преимущество Гибридного алгоритма.

Можно сделать вывод, что построение "смешанных" гибридных алгоритмов дает хорошие результаты.

9.8 Алгоритм Муравьиные Колонии

В данной разделе рассматривается алгоритм, основанный на методе Муравьиные Колонии (Ant Colony Optimization. ACO). Этот итерационный алгоритм (далее – *алгоритм ACO*) основан на идее последовательного приближения к оптимальному решению.

Каждая итерация – "запуск искусственного муравья – который "пытается" по некоторому правилу выбрать наилучший маршрут к "пище" (к оптимуму функции), используя метки своих предшественников.

Каждый муравей выполняет цепочку шагов. На каждом шаге i, $i = 1, 2, \ldots, n$, выбирается требование j для постановки на место i в расписании, используя "вероятности перехода".

В алгоритме используются параметры:

η_{ij} – эвристическая информация о том, насколько хорошим кажется постановка требования j на место i в расписании. Этот параметр вычисляется эвристически по одному из вариантов:

1. По правилу EDD: $\eta_{ij} = \frac{1}{d_j},\ i = 1, \ldots, n$;
2. По правилу MDD (modified due date). В алгоритме MDD последовательно на позиции $i = 1, \ldots, n$ выбирается еще неупорядоченное

требование j с наименьшим значением $max\{S+p_j, d_j\}$, где S – сумма продолжительностей предшествующих упорядоченных требований. Эвристическая информация рассчитывается следующим образом: $\eta_{ij} = \frac{1}{max\{S+p_j,d_j\}}$;

3. По правилу L-MDD (Look-ahead MDD): $\eta_{ij} = \frac{1}{Tard_j}$, где $Tard_j$ – суммарное запаздывание при модифицированном расписании MDD, при котором на позиции i обслуживается требование j, а все остальные требования упорядочены в соответствии с правилом MDD;
4. По правилу SPT: $\eta_{ij} = \frac{1}{p_j},\ i = 1, \ldots, n$;

τ_{ij} – "след" (в природе: след феромона). После каждой итерации этот параметр корректируется. Параметр показывает насколько "хорошим" для требования j оказалась позиция i. То есть это накопленная статистическая информация о качестве выбора для позиции i требования j, в то время как η_{ij} характеризует предполагаемую выгоду такой постановки при недостатке накопленной информации.

Перед первой итерацией полагают $\tau_{ij} = 1/(mT_{EDD})$, где T_{EDD} – суммарное запаздывание при EDD-расписании. m–количество итераций (муравьев). Параметры η_{ij} рассчитываются один раз перед первой итерацией.

На каждом шаге $i,\ i = 1, 2, \ldots, n$, вычисляется *Матрица вероятностей перехода*:

$$\rho_{ij} = \begin{cases} \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{h\in\Omega}[\tau_{ih}]^\alpha [\eta_{ih}]^\beta},\ j \in \Omega, \\ 0\ ,\ j \notin \Omega, \end{cases}$$

где Ω – множество неупорядоченных требований.

Правило, по которому на позицию i выбирается требование j определяется следующим образом:

$$\begin{cases} j = argmax_{h\in\Omega}[\tau_{ih}]^\alpha [\eta_{ih}]^\beta, & q < q_0, \\ j \text{ определяется случайным образом,} \\ \text{согласно распределению вероятностей, } \rho_{ij}, & q \geq q_0, \end{cases}$$

где $0 \leq q_0 \leq 1$ – параметр алгоритма, а значение q вычисляется случайным образом на каждом шаге.

После того, как требование j было поставлено на позицию i, пересчитывается "локальный след":

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho\tau_0,$$

где $\tau_0 = 1/(mT_{EDD})$, а $\rho \in [0,1]$ – коэффициент распада феромона (параметр алгоритма).

После каждой итерации "глобальный след" τ_{ij} корректируется по правилу:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho/T^*,$$

если в "лучшем" найденном расписании на позиции i обслуживается требование j. Иначе

$$\tau_{ij} = (1-\rho)\tau_{ij},$$

Значение T^* – суммарное запаздывание для "лучшего" найденного расписания.

Алгоритм ACO дает хорошие результаты при интеграции в него локального поиска, к примеру, попарной перестановки требований. Локальный поиск запускается после каждой итерации.

Нетрудно убедиться, что трудоемкость алгоритма без локального поиска составляет $O(mn^2)$ операций. Для каждой позиции i (всего n позиций) выбиралось требование j за $O(n)$ операций.

Трудоемкость локального поиска $O(n^3)$. Тогда теоретическая трудоемкость алгоритма ACO с локальным поиском составляет не меньше $O(mn^3)$ операций.

9.9 Гибридный алгоритм решения

На основе Алгоритма ACO и точного алгоритма 9.3 с правилами исключения 1-3 нами построен новый *Гибридный алгоритм*. Идея алгоритма заключается в том, что на каждой итерации запускается модифицированный алгоритм 9.3, в котором очередное требование j^* "случайным" образом ставится на некоторую позицию $k \in L(N,t)$.

"Случайный" выбор происходит согласно методу Муравьиные колонии, т.е. подзадача выбора подходящей позиции решается Алгоритмом ACO. Модифицированная процедура ProcL, используемая в алгоритме, представлена далее (см. блок алгоритма 9.4).

После каждой итерации "глобальный след" τ_{ij} корректируется по правилу:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho/T^*,$$

Алгоритм 9.4 Модифицированная процедура ProcL

Procedure ProcL (N, t)

1: Дан пример $\{N, t\}$ с множеством требований $N = \{j_1, j_2, \ldots, j_n\}$ и моментом начала обслуживания t, где $d_{j_1} \leq d_{j_2} \leq \ldots \leq d_{j_n}$.
2: **if** $N = \emptyset$ **then**
3: π^*:= пустое расписание;
4: **else**
5: Найдем требование j^* из множества N;
6: Найдем множество $L(N, t)$ для требования j^*;
7: Рассчитаем матрицу вероятностей перехода для каждой позиции $i \in L(N, t)$.

$$\rho_{ij*} = \frac{\tau_{ij*}/F(\pi^i, t)}{\sum_{h \in L(N,t)} \tau_{hj*}/F(\pi^h, t)},$$

где $\pi^i = (j_1, \ldots, j_{m-1}, j_{m+1}, \ldots, j_i, j^*, j_{i+1}, \ldots, j_n)$, $d_{j_1} \leq \ldots, d_{j_n}$, $j^* = j_m, m < i$, $F(\pi^i, t)$ – суммарное запаздывание при расписании π^i, построенном с момента времени t.
8: Выберем позицию $k \in L(N, t)$ произвольным образом согласно распределению вероятностей ρ_{ij*}.
9: Пересчитаем "локальный след":

$$\tau_{kj*} = (1 - \rho)\tau_{kj*} + \rho\tau_0,$$

где $\tau_0 = 1/(mT_{EDD})$. T_{EDD} – суммарное запаздывание при EDD-расписании.
10: $\pi_k := (\mathbf{ProcL}(N', t'), j^*(N), \mathbf{ProcL}(N'', t''))$, где
$N' := \{j_1, \ldots, j_k\} \setminus \{j^*\}$, $t' := t$, $N'' := \{j_{k+1}, \ldots, j_n\}$, $t'' := t + \sum_{i:=1}^{k} p_{j_i}$;
11: **end if**
12: **RETURN** π^*;

если в "лучшем" найденном расписании на позиции i обслуживается требование j. Иначе

$$\tau_{ij} = (1 - \rho)\tau_{ij},$$

где $\rho \in [0, 1]$ – параметр алгоритма. Значение T^* – суммарное запаздывание для "лучшего" найденного расписания.

После каждой итерации запускается процедура локального поиска – попарная перестановка требований.

Нетрудно убедиться, что трудоемкость Гибридного алгоритма без локального поиска $O(mn^2)$ операций. Процедура ProcL запускается n раз. В каждой процедуре выполняется порядка $O(n)$ операций.

Трудоемкость локального поиска $O(n^3)$. Тогда теоретическая трудоемкость Гибридного алгоритма с локальным поиском составляет не меньше $O(mn^3)$ операций. То есть трудоемкость идентична трудоемкости алгоритма ACO.

9.10 Минимизация обобщенной функции запаздывания

В предыдущих разделах мы познакомились с двумя задачами, целевые функции которых связаны с запаздыванием требований $1||\sum w_jU_j$ и $1||\sum T_j$. В этом разделе приводится обобщение этих двух задач.

В дополнении к обычным параметрам p_j и d_j, для каждого требования j, $j = 1, 2, \ldots, n$, заданы *квота запаздывания* $b_j \geq 0$, *коэффициент нормальноого запаздывания* $v_j \geq 0$ и *коэффициент ненормальноого запаздывания* $w_j \geq 0$.

Обобщенное запаздывание задается следующим образом:

$$GT_j(\pi) = \begin{cases} 0, & \text{если} C_j(\pi) - d_j \leq 0, \\ v_j \cdot (C_j(\pi) - d_j), & \text{если} 0 < C_j(\pi) - d_j \leq b_j, \\ w_j, & \text{если} b_j < C_j(\pi) - d_j, \end{cases}$$

где $w_j \geq v_j b_j$ для каждого $j \in N$. Целевая функция задачи:

$$F(\pi) = \sum_{j=1}^{n} GT_j(\pi) \to \min .$$

Функцию обобщённого запаздывания можно интерпретировать следующим образом. Если запаздывание требования j превышает параметр b_j, то значение штрафа уже не зависит от запаздывания, остается постоянным и равным w_j. Необходимо найти расписание, минимизирующее значение $F(\pi)$. Эту задачу будем обозначать как $1||\sum GT_j$.

Очевидно, что эта задача NP-трудна, так как ее частный случай $b_j = 0$ соответствует NP-трудной проблеме $1||\sum w_jU_j$.

В данном разделе мы рассмотрим частный случай задачи, при котором

$$b_j = p_j, \quad v_j = 1, \quad w_j = p_j, \tag{9.2}$$

то есть $GT_j(\pi) = \min\{\max\{0, C_j(\pi) - d_j\}, p_j\}$ для всех $j \in N$.

Теорема 9.9 *Частный случай (9.2) задачи $1||\sum GT_j$ является NP-трудным.*

Доказательство. Сведем к данному частному случаю ЗАДАЧУ РАЗБИЕНИЯ. Дан пример ЗАДАЧИ РАЗБИЕНИЯ. Построим пример частного случая (9.2) с $n + 1$ требованиями. Продолжительности обслуживания $p_j = b_j$, $j = 1, 2, \ldots, n$, и $p_{n+1} = 1$, т.е. $\sum\limits_{j=1}^{n+1} p_j = 2A + 1$. Директивные сроки $d_j = A$,

$j = 1, 2, \ldots, n$, и $d_{n+1} = A + 1$. При любом расписании π выполняется $F(\pi) \geq A = \sum\limits_{j=1}^{n+1} p_j - d_{n+1}$. Более того, для всех расписаний выполняется $F(\pi) \leq A + 1$.

При оптимальном расписании $\pi^* = (\pi_1, n+1, \pi_2)$ (без простоев) равенство $C_{n+1}(\pi^*) = A + 1$ выполняется тогда и только тогда, когда ответ в примере ЗАДАЧИ РАЗБИЕНИЯ – "ДА". Все требования из частичного расписания π_1 и требование $n+1$ не запаздывают, а требования из π_2 запаздывают. Требование из частичного расписания π_1 соответствуют элементам найденного подмножества чисел N' примера ЗАДАЧИ РАЗБИЕНИЯ. Причем, $F(\pi^*) = A$. Таким образом задачи (9.2) $1\|\sum GT_j$ и РАЗБИЕНИЯ являются эквивалентными, что и доказывает NP-трудность частного случая (9.2). $\square$

Лемма 9.5 *Для частного случая (9.2) существует оптимальное расписание вида $\pi = (G, H)$, где для всех требований $i \in G$ выполняется $0 \leq$ $\leq GT_i(\pi) < p_i$, а для требований $j \in H$ выполняется $GT_i(\pi) = p_i$. Требования из множества G обслуживаются в порядке EDD[1], а требования из множества H – в порядке LDD[2].*

Доказательство. 1) Допустим, что существует оптимальное расписание вида $\pi^* = (\pi_1, j, \pi_2)$. Если $GT_j(\pi) = p_j$, тогда расписание $\pi' = (\pi_1, \pi_2, j)$ также оптимальное. То есть существует оптимальное расписание вида $\pi = (G, H)$, где все требования $j \in H$ запаздывают и $GT_j(\pi) = p_j$. Для всех требований $i \in G$ выполняется $0 \leq GT_i(\pi) < p_i$.

2) Рассмотрим оптимальное расписание вида $\pi = (G, H)$, где все требования $j \in H$ запаздывают и $GT_j(\pi) = p_j$, а для всех требований $i \in G$ выполняется $0 \leq GT_i(\pi) < p_i$. Докажем, что требования $i \in G$ обслуживаются в порядке EDD.

Предположим, что существует оптимальное расписание вида $\pi = (\pi_1, \alpha, \beta, \pi_2)$, где требования $\alpha, \beta \in G$, и выполняется $d_\alpha > d_\beta$. А также выполняется $C_\alpha(\pi) - d_\alpha < p_\alpha$ и $C_\beta(\pi) - d_\beta < p_\beta$, тогда $C_\alpha(\pi) < d_\alpha$.

Рассмотрим расписание $\pi' = (\pi_1, \beta, \alpha, \pi_2)$. Определим $C = C_\beta(\pi) = C_\alpha(\pi')$. Тогда

$$F(\pi) - F(\pi') = (GT_\alpha(\pi) - GT_\alpha(\pi')) + (GT_\beta(\pi) - GT_\beta(\pi')) =$$

$$= -\min\{p_\alpha, \max\{0, C - d_\alpha\}\} + \min\{p_\alpha, \max\{0, C - d_\beta\}\} \geq 0$$

[1]EDD—earliest due date
[2]LDD—lates due date

и расписание π' также оптимально.

3) Докажем, что требования $j \in H$ обслуживаются в порядке LDD. Для всех требований $j \in H$ имеем $d_j \leq \sum\limits_{l=1}^{n} p_l - \sum\limits_{k \in H} p_k$, иначе, если $d_j > \sum\limits_{l=1}^{n} p_l - \sum\limits_{k \in H} p_k$, тогда при расписании $\pi' = (G, j, H \setminus \{j\})$ мы имеем меньшее значение целевой функции, т.е. получили противоречие, т.к. π – оптимальное расписание. Поэтому требования из H могут быть обслужены в любом порядке. □

Пользуясь этой леммой, можно построить точный алгоритм решения данного частного случая. Трудоёмкость построенного алгоритма будет равна $O(nd_{\max})$ операций, где $d_{\max}$ – максимальный директивный срок.

9.11 Одноприборные задачи с обратными критериями оптимизации

Представьте себе, что нам необходимо максимизировать суммарное запаздывание или максимизировать число запаздывающих требований, в отличие от классических задач, где эти значения нужно минимизировать. Несмотря на абсурдность такой максимизации, данные задачи представляют собой теоретический и практический интерес. В данном разделе рассматриваются одноприборные задачи с "обратными" критериями оптимальности, например, задачи максимизации суммарного запаздывания и максимизации количества запаздывающих требований для одного прибора.

Формулируются эти задачи следующим образом, практически совпадающим с формулировками классических задач минимизации.

Необходимо обслужить n требований на одном приборе. Прерывания при обслуживании и обслуживание более одного требования в любой момент времени запрещены. Для каждого требования $j \in N = \{1, 2, \ldots, n\}$ заданы продолжительность обслуживания $p_j > 0$ и директивный срок его окончания d_j, где N – множество требований, которые необходимо обслужить. Прибор начинает обслуживание требований с момента времени 0. Простои прибора при обслуживании требований запрещены (иначе задачи максимизации становятся тривиальными). Расписание обслуживания требований $\pi = (j_1, j_2, \ldots, j_n)$ строится с момента времени 0 и однозначно задаётся перестановкой элементов множества N. Обозначим через $C_{j_k}(\pi) = \sum\limits_{l=1}^{k} p_{j_l}$ время завершения обслуживания требований j_k при расписании π. Если $C_j(\pi) > d_j$, тогда требование j запаздывает, и в этом случае полагают $U_j = 1$. Если $C_j(\pi) \leq d_j$, тогда требование j не запаздывает, и $U_j = 0$

Требуется построить расписание π^* обслуживания требований множества N, при котором достигается максимум функции $F(\pi) = \sum_{j=1}^{n} U_j(\pi)$. Обозначим данную задачу через $1(nd)||\max\sum U_j$. Аналогично, обозначается задача максимизации суммарного запаздывания $1(nd)||\max\sum T_j$ и другие задачи максимизации.

Запись (nd) означает, что рассматриваются расписания, при которых прибор не простаивает (non-delay), т.е. все требования обслуживаются в интервале $[0, \sum p_j]$ без перерывов. В литературе можно встретить и другие задачи максимизации, обозначаемые символами $1(sa)|\ldots|\max\ldots$, в которых рассматриваются т.н. semi-active расписания.

В этом главе приводятся доказательства NP-трудности некоторых задач максимизации, а также полиномиальный графический алгоритм решения задачи $1(nd)||\max\sum T_j$.

С одной стороны, исследование данных задач само по себе является важной теоретической задачей. Алгоритмы решения данных задач могут быть использованы для вычисления верхних оценок, исследования свойств оптимальных расписаний, вычисления частичного порядка обслуживания требований для "исходных" минимизационных задач, а также для сокращения перебора в алгоритмах решения "исходных" задач. Например, алгоритм решения задачи максимизации количества запаздываюших требований $1||\max\sum U_j$ используется для вычисления максимального количества запаздываюших требований, что, в свою очередь, позволяет сократить перебор в алгоритме решения задачи максимизации суммарного запаздывания $1||\max\sum T_j$. Значение $\max\sum T_j$ может быть использовано при решении классической задачи минимизации $1||(\alpha\sum E_j + \beta\sum T_j)$. Например, если $\alpha > \beta$, тогда можно вычислить максимальное значение $\beta\sum T_j$ и после искать расписание, оптимальное только с точки зрения критерия $\sum E_j$.

Также теоретический интерес представляет корреляция между полиномиально разрешимыми и NP-трудными случаями для "исходной" и "обратной" задач.

С другой стороны, для данных проблем существуют практические интерпретации и приложения. Например, монтажная команда должна смонтировать ветряные электрогенераторы (турбины) в разных районах страны. В каждом районе j необходимо смонтировать определенное количество турбин. Время монтажа p_j зависит только от количества турбин и не зависит от погодных или климатических условий. Однако погода влияет на дополнительные расходы (например, на расход топлива, на зарплату и стоимость

Таблица 9.4: Сведения о трудоёмкости задач максимизации

Задача	трудоёмкость и алгоритмы	трудоёмкость соотв. задачи минимизации
$1(nd)\|\|\max\sum U_j$	Полин. разрешима за время $O(n\log n)$	Полин. разрешима за время $O(n\log n)$
$1(nd)\|D_j\|\max\sum T_j$	NP-трудна	???
$1(nd)\|D_j\|\max\sum U_j$	NP-трудна	???
$1(nd)\|D_j\|\max\sum w_jC_j$	NP-трудна	???
$1(nd)\|D_j\|\max\sum C_j$	NP-трудна	???
$1(nd)\|\|\max\sum w_jU_j$	NP-трудна	NP-трудна
$1(nd)\|r_j\|\max\sum U_j$	NP-трудна	NP-трудна
$1(nd)\|r_j\|\max\sum w_jC_j$	NP-трудна	NP-трудна
$1(nd)\|r_j\|\max\sum C_j$	Полин. разрешима за время $O(n\log n)$	NP-трудна
$1(nd)\|prec,r_j\|\max C_{\max}$	$O(n^2)$	NP-трудна
$1(nd)\|prec,r_j\|\max f_{\max}$	$O(n^4)$	NP-трудна
$1(nd)\|r_j\|\max f_{\max}$	$O(n^3)$	NP-трудна
$1(nd)\|prec,r_j\|\max\sum C_j$	NP-трудна	NP-трудна
$1(nd)\|prec,r_j\|\max\sum U_j$	NP-трудна	NP-трудна
$1(nd)\|\|\max\sum w_jT_j$	NP-трудна, Алгоритм решения трудоёмкости $O(n\min\{\sum w_j,d_{\max}\})$	NP-трудна
$1(nd)\|r_j\|\max\sum w_jT_j$	NP-трудна	NP-трудна
$1(nd)\|r_j\|\max\sum T_j$	NP-трудна	NP-трудна
$1(nd)\|\|\max\sum T_j$	Алгоритм решения трудоёмкости $O(n\sum p_j)$. Алгоритм решения трудоёмкости $O(n^2)$	NP-трудна. Алгоритм решения трудоёмкости $O(n^4\sum p_j)$
$1(sa)\|r_j\|\max\sum T_j$	Полин.разрешима за время $O(n^3)$	
$1(sa)\|r_j\|\max\sum w_jT_j$	NP-трудна	

проживания рабочих, которые могут быть выше зимой). Сумма этих дополнительных расходов начинает быстро снижаться после схода снега. Для каждого региона (т.е. для каждого требования) дан прогноз, когда ожидается сход снега, т.е. когда снег растает (этот момент времени можно интерпретировать как директивный срок). Целевая функция – минимизировать эти дополнительные расходы, т.е. целевая функция может быть интерпретирована как $\max\sum\max\{0,S_j-d'_j\}$, где $S_j=C_j-p_j$ и $d'_j=d_j-p_j$. В результате получили задачу $1||\max\sum T_j$.

Сведения о трудоёмкости задач максимизации сведены в таблицу 9.4.

Большинство результатов для задач максимизации, представленных в таблице, могут быть получены без особых усилий. Далее приведены три про-

стых доказательства NP-трудности некоторых из перечисленных задач.

Лемма 9.6 *Задачи* $1(nd)|D_j|\max\sum T_j$ *и* $1(nd)|D_j|\max\sum U_j$ *NP-трудны.*

Доказательство. Доказательство проведем сведением ЗАДАЧИ РАЗБИЕНИЯ к данным задачам. Дан пример ЗАДАЧИ РАЗБИЕНИЯ с n числами. Конструируем пример исследуемых задач следующим образом.
В примере $n+1$ требование, где $p_j = b_j$ и $d_j = D_j = \sum\limits_{j=1}^{n} p_j + 1$, $j = 1, 2, \ldots, n$, то есть при любом допустимом расписании эти требования не запаздывают.
Пусть $p_{n+1} = 1$, $d_{n+1} = A$, $D_{n+1} = A + 1$, где $A = \frac{1}{2}\sum\limits_{j=1}^{n} p_j$.

Если ответ в примере ЗАДАЧИ РАЗБИЕНИЯ "ДА",
тогда существует оптимальное расписание $\pi = (\pi_1, n+1, \pi_2)$, где
$\{\pi_1\} = N'$, $\sum\limits_{j\in N'} p_j = A$, $\{\pi_2\} = N \setminus N'$ и $\sum\limits_{j=1}^{n+1} T_j(\pi) = 1$.
Если ответ "НЕТ", тогда для всех допустимых расписаний $\sum\limits_{j=1}^{n+1} T_j = 0$.
Соответственно, выполняются $\sum\limits_{j=1}^{n+1} U_j = 1$ и $\sum\limits_{j=1}^{n+1} U_j = 0$. □

Лемма 9.7 *Задачи* $1(nd)|r_j|\max\sum w_jC_j$ *и* $1(nd)|D_j|\max\sum w_jC_j$ *NP-трудны.*

Доказательство. Доказательство проведем сведением ЗАДАЧИ РАЗБИЕНИЯ к данным задачам. Дан пример ЗАДАЧИ РАЗБИЕНИЯ с n числами. Конструируем пример задачи $1(nd)|r_j|\max\sum w_jC_j$ следующим образом. В примере $n+1$ требование, где $p_j = w_j = b_j$ и $r_j = 0$, $j = 1, 2, \ldots, n$.
Пусть $p_{n+1} = 1$, $w_{n+1} = 0$, $r_{n+1} = A$, где $A = \frac{1}{2}\sum\limits_{j=1}^{n} p_j$.

Если ответ в примере ЗАДАЧИ РАЗБИЕНИЯ "ДА",
тогда существует оптимальное расписание $\pi = (\pi_1, n+1, \pi_2)$,
где $\{\pi_1\} = N'$, $\sum\limits_{j\in\pi_1} p_j = A$, $\{\pi_2\} = N \setminus N'$, $\sum\limits_{j\in\pi_2} p_j = \sum\limits_{j\in N} b_j - A$ и

$$\sum_{j=1}^{n+1} w_jC_j(\pi) = \sum_{1\le i\le j\le n} b_ib_j + \left(\sum_{j\in N} b_j - A\right),$$

так как при расписании $\pi' = (\pi_1, \pi_2, n+1)$ выполняется

$$\sum_{j=1}^{n+1} w_j C_j(\pi) = \sum_{1 \le i \le j \le n} b_i b_j.$$

Требования в частичных расписаниях π_1 и π_2 могут быть обслужены в любом порядке.
Если ответ "НЕТ", то

$$\sum_{j=1}^{n+1} w_j C_j(\pi) < \sum_{1 \le i \le j \le n} b_i b_j + \left(\sum_{j \in N} b_j - A \right).$$

Следовательно частный случай задачи $1(nd)|r_j|\max \sum w_j C_j$ эквивалентен NP-трудной задаче РАЗБИЕНИЯ, поэтому задача $1(nd)|r_j|\max \sum w_j C_j$ является NP-трудной.

Аналогично для задачи $1(nd)|D_j|\max \sum w_j C_j$ мы конструируем пример с множеством из $n+1$ требований, где $p_j = b_j,\ w_j = 0,\ D_j = \sum\limits_{i=1}^{n+1} p_i,\ j = 1, 2, \ldots, n$, а также $p_{n+1} = 1,\ w_{n+1} = 1$ и $D_{n+1} = A + 1$.

Ответ для примера ЗАДАЧИ РАЗБИЕНИЯ "ДА" тогда и только тогда, когда при оптимальном расписании π выполняется $C_{n+1}(\pi) = A + 1$. □

Лемма 9.8 *Задача* $1(nd)|D_j|\max \sum C_j$ *NP-трудна.*

Доказательство. Очевидно, что две задачи $1|r_j|\min \sum C_j$ и $1|r_j|\min \sum S_j$ эквивалентны, т.к. $\sum\limits_{j \in N} C_j = \sum\limits_{j \in N} S_j + \sum\limits_{j \in N} p_j$. Известно, что задача $1|r_j|\min \sum C_j$ NP-трудна. Легко показать, что задачи $1(nd)|D_j|\max \sum C_j$ и $1|r_j|\min \sum S_j$ также эквивалентны.

Пусть имеется пример задачи $1|r_j|\min \sum S_j$ с множеством N из n требований. Для примера задачи $1(nd)|D_j|\max \sum C_j$, зададим n требований с параметрами p'_j, D'_j, определенными по правилам:
$p'_j = p_j,\ D'_j = \sum\limits_{j \in N} p_j - r_j, j = 1, 2, \ldots, n.$
Рассмотрим расписание $\pi = (j_1, j_2, \ldots, j_k, j_{k+1}, \ldots, j_n)$.
Определим $H_{j_k} = \sum\limits_{i=k+1}^{n} p_{j_i}$. Тогда задача максимизации $\sum\limits_{j \in N} C_j = n \sum\limits_{j \in N} p_j - \sum\limits_{j \in N} H_j$ сводится к задаче минимизации $\sum\limits_{j \in N} H_j$, то есть сводится к задаче $1|r_j|\min \sum S_j$.

Пусть $\pi = (1, 2, \ldots, n)$ – оптимальное расписание для примера задачи $1|r_j|\min\sum S_j$. Тогда расписание $\pi' = (n, \ldots, 2, 1)$ будет оптимальным для соответствующего примера задачи $1|D_j|\max\sum C_j$.

Поэтому две проблемы эквивалентны и, следовательно, исследуемая задача $1(nd)|D_j|\max\sum C_j$ является NP-трудной. □

9.11.1 Доказательство NP-трудности задачи $1(nd)||\max\sum w_jT_j$

Далеко не все доказательства NP-трудности для задач теории расписаний столь тривиальны. Для иллюстрации нетрививиального доказательства рассмотрим задачу $1(nd)||\max\sum w_jT_j$. Доказательство проведем сведением ЗАДАЧИ РАЗБИЕНИЯ к частному случаю задачи $1(nd)||\max\sum w_jT_j$. Без потери общности, пусть в примере ЗАДАЧИ РАЗБИЕНИЯ $n > 3$ и $\sum\limits_{i=1}^{n} b_i > 10$.

Дан произвольный пример ЗАДАЧИ РАЗБИЕНИЯ, мы конструируем следующий пример задачи $1(nd)||\max\sum w_jT_j$:

$$\begin{cases} w_{2i} = M^i,\ i = 1, 2, \ldots, n, & (3.3.1) \\ w_{2i-1} = w_{2i} + b_i,\ i = 1, 2, \ldots, n, & (3.3.2) \\ p_{2i} = \sum\limits_{j=1}^{i-1} w_{2j} + \frac{1}{2}\sum\limits_{j\in N\setminus\{i\}} b_j,\ i = 1, 2, \ldots, n, & (3.3.3) \\ p_{2i-1} = p_{2i} + b_i,\ i = 1, 2, \ldots, n, & (3.3.4) \\ d_{2i} = d_{2i-1} = P - \sum\limits_{j=i}^{n} p_{2j},\ i = 1, 2, \ldots, n, & (3.3.5) \end{cases} \tag{9.3}$$

где $M = (n\sum\limits_{i=1}^{n} b_i)^{10}$ и $P = \sum\limits_{j=1}^{2n} p_j$.

Обозначим работы из множества $\bar{N} = \{1, 2, \ldots, 2n\}$ следующим образом:

$$V_1, V_2, V_3, V_4, \ldots, V_{2i-1}, V_{2i}, \ldots, V_{2n-1}, V_{2n}.$$

Каноническим расписанием будем называть расписание вида

$$(V_{n,1}, V_{n-1,1}, \ldots, V_{i,1}, \ldots, V_{1,1}, V_{1,2}, \ldots, V_{i,2}, \ldots, V_{n-1,2}, V_{n,2}),$$

где $\{V_{i,1}, V_{i,2}\} = \{V_{2i-1}, V_{2i}\}$, $i = 1, 2, \ldots, n$.

Лемма 9.9 *Для каждого примера задачи* $1(nd)||\max\sum w_jT_j$ *существует оптимальное расписание вида* $\pi = (G, H)$, *где все требования* $j \in H$ *запаздывают, а все требования* $i \in G$ *не запаздывают. Все требования из множества* G *обслуживаются в порядке невозрастания значений* $\frac{w_j}{p_j}$, *а все требования из множества* H *обслуживаются в порядке неубывания значений* $\frac{w_j}{p_j}$.

Доказательство. 1) Предположим, что существует оптимальное расписание вида $\pi = (\pi_1, j, \pi_2, i, \pi_3)$, при котором требование j запаздывает, а требование i не запаздывает. Для расписания $\pi' = (\pi_1, i, j, \pi_2, \pi_3)$ выполняется: $F(\pi') - F(\pi) \geq w_j(T_j(\pi') - T_j(\pi)) + w_i(T_i(\pi') - T_i(\pi)) = w_j(p_i) + (0) > 0$. Получили противоречие, так как для расписания π' значение целевой функции больше, а значит расписание π не оптимальное. Повторив необходимое число раз подобное преобразование расписания, мы получим оптимальное расписание вида $\pi = (G, H)$.

2) Рассмотрим оптимальное расписание вида $\pi = (G, H)$, где все требования $j \in H$ запаздывают, а все требования $i \in G$ не запаздывают. Покажем, что все требования $j \in H$ обслуживаются в порядке неубывания значений $\frac{w_j}{p_j}$. Предположим, что существует оптимальное расписание $\pi = (\pi_1, j_1, j_2, \pi_2)$, при котором требования j_1 и j_2 запаздывают и $\frac{w_{j_1}}{p_{j_1}} > \frac{w_{j_2}}{p_{j_2}}$, т.е. $w_{j_1}p_{j_2} > w_{j_2}p_{j_1}$. Для расписания $\pi' = (\pi_1, j_2, j_1, \pi_2)$ выполняется $F(\pi') - F(\pi) = w_{j_1}(T_{j_1}(\pi') - - T_{j_1}(\pi)) + w_{j_2}(T_{j_2}(\pi') - T_{j_2}(\pi)) \geq w_{j_1}p_{j_2} - w_{j_2}\min\{p_{j_1}, T_{j_2}(\pi)\} > 0$. Получили противоречие, а значит, расписание $\pi = (\pi_1, j_1, j_2, \pi_2)$ не оптимальное. После необходимого количества преобразований расписания π мы получим оптимальное расписание, при котором все требования множества H будут упорядочены в порядке неубывания величин $\frac{w_j}{p_j}$.

3) Рассмотрим оптимальное расписание вида $\pi = (G, H)$, где все требования $j \in H$ запаздывают, а все требования $i \in G$ не запаздывают. Покажем что все требования $i \in G$ могут обслуживаться в порядке невозрастания значений $\frac{w_i}{p_i}$ при оптимальном расписании. Для всех требований $i \in G$ выполняется $d_i \geq \sum\limits_{k\in G} p_k$, иначе, если $d_i < \sum\limits_{k\in G} p_k$, то расписание $\pi' = (G \setminus \{i\}, i, H)$ "лучше" (т.е. для данного расписания значение целевой функции больше), а следовательно, получено противоречие. Поэтому все требования $i \in G$ могут быть обслужены в любом порядке (в том числе, и в порядке невозрастания величин $\frac{w_i}{p_i}$), так как при любом порядке обслуживания все требования из G не запаздывают. □

Следствием из этой леммы является следующее утверждение.

Лемма 9.10 *Для каждого примера задачи* $1(nd)||\max\sum T_j$ *существует оптимальное расписание вида* $\pi = (G, H)$*, где все требования* $j \in H$ *запаздывают, а все требования* $i \in G$ *не запаздывают. Все требования из множества* G *обслуживаются в порядке неубывания продолжительности обслуживания (SPT), а все требования из множества* H *обслуживаются в порядке невозрастания продолжительности обслуживания (LPT* [3]*).*

□

Лемма 9.11 *Для частного случая (9.3) все оптимальные расписания являются каноническими или могут быть сведены к каноническим расписаниям, если упорядочить первые* n *требований в расписании по правилу LPT (longest processing time).*

Доказательство. 1) Сначала докажем, что одно из двух требований, V_{2n} или V_{2n-1}, запаздывает при любом оптимальном расписании. Предположим, что оба требования не запаздывают при некотором оптимальном расписании $\pi = (\pi_1, V_{2n}, \pi_2, V_{2n-1}, \pi_3, \pi_4)$, где запаздывают только требования из частичного расписания π_4. Мы рассматриваем только оптимальные расписания вида $\pi = (G, H)$, где все требования $j \in H$ запаздывают, а все требования $i \in G$ не запаздывают (см. лемму 9.9). Для расписания $\pi' = (\pi_1, \pi_2, V_{2n-1}, \pi_3, \pi_4, V_{2n})$ имеем:

$$\sum_{j=1}^{2n} w_j T_j(\pi') - \sum_{j=1}^{2n} w_j T_j(\pi) \geq w_{2n} T_{2n}(\pi') - p_{2n} \sum_{j \in \{\pi_4\}} w_j \geq$$

$$\geq p_{2n} M^n - p_{2n} \sum_{i=1}^{2n-1} (2M^i + b_i) = p_{2n} M^n - p_{2n} \left(2M \frac{M^{n-1} - 1}{M - 1} + \sum_{i=1}^{2n-1} b_i \right) > 0.$$

Значит расписание π не оптимальное, т.е. одно и только одно из двух требований, V_{2n} или V_{2n-1}, запаздывает при любом оптимальном расписании. **2)** Докажем верность следующего неравенства: существует оптимальное расписание, при котором

$$\frac{w_2}{p_2} < \frac{w_4}{p_4} < \ldots < \frac{w_{2n}}{p_{2n}}.$$

[3] Longest Processing Time

Необходимо доказать, что

$$\frac{M^{i-1}}{\sum\limits_{j=1}^{i-2} w_{2j} + \frac{1}{2}\sum\limits_{j\in N\setminus\{i-1\}} b_j} < \frac{M^{i}}{\sum\limits_{j=1}^{i-1} w_{2j} + \frac{1}{2}\sum\limits_{j\in N\setminus\{i\}} b_j}\ .$$

Обозначим $B_i = \frac{1}{2}\sum\limits_{j\in N\setminus\{i\}} b_j,\ i = 1, 2, \ldots, n$. Тогда мы имеем:

$$\frac{M^{i-1}}{M\frac{M^{i-2}-1}{M-1} + B_{i-1}} < \frac{M^{i}}{M\frac{M^{i-1}-1}{M-1} + B_{i}}$$

$$\Longleftrightarrow \frac{1}{\frac{M(M^{i-2}-1)+B_{i-1}(M-1)}{M-1}} < \frac{M}{\frac{M(M^{i-1}-1)+B_{i}(M-1)}{M-1}}$$

$$\Longleftrightarrow M(M^{i-1}-1) + B_i(M-1) < M[M(M^{i-2}-1) + B_{i-1}(M-1)]$$

$$\Longleftrightarrow 0 < M^2(B_{i-1}-1) - M(B_{i-1}+B_i-1) + B_i, \quad i = 1, 2, \ldots, n.$$

Последнее неравенство истинно, т.к. $M^2 > M \cdot 2\sum\limits_{j=1}^{n} b_j$ и $(B_{i-1}-1) > 1$. То есть исходное неравенство верно. Аналогично можно доказать, что

$$\frac{w_{2(i-1)-1}}{p_{2(i-1)-1}} < \frac{w_{2i}}{p_{2i}}, \quad \frac{w_{2(i-1)}}{p_{2(i-1)}} < \frac{w_{2i-1}}{p_{2i-1}} \quad \text{и} \quad \frac{w_{2(i-1)-1}}{p_{2(i-1)-1}} < \frac{w_{2i-1}}{p_{2i-1}}$$

для любого $i = 2, 3, \ldots, n$.

Тогда одно из двух требований V_{2n} или V_{2n-1} является **последним** запаздывающим требованием при любом оптимальном расписании (см. лемму 9.9). В дальнейшем будем рассматривать только расписания вида $(V_{n,1}, \pi_\alpha, V_{n,2})$, где $\{V_{n,1}, V_{n,2}\} = \{V_{2n-1}, V_{2n}\}$.

3) Аналогично доказательствам из пп. 1) и 2), можно доказать, что для каждого $i = n-1, n-2, \ldots, 1$, одно из двух требований V_{2i} или V_{2i-1} запаздывает при любом оптимальном расписании. □

Теорема 9.10 *Задача* $1(nd)||\max\sum w_jT_j$ *NP-трудна в обычном смысле.*

Доказательство. Для расписания π вида

$$\pi = (V_{2n-1}, V_{2(n-1)-1}, \ldots, V_3, V_1, V_2, V_4, \ldots, V_{2(n-1)}, V_{2n})$$

имеем следующее значение целевой функции:

$$F(\pi) = \sum_{j=1}^{n} w_{2j} T_{2j}(\pi) = \sum_{j=1}^{n} w_{2j} p_{2j}.$$

Рассмотрим каноническое расписание вида

$$\pi' = (V_{n,1}, V_{n-1,1}, \ldots, V_{i,1}, \ldots, V_{1,1}, V_{1,2}, \ldots, V_{i,2}, \ldots, V_{n-1,2}, V_{n,2}).$$

Определим переменную

$$x_i = \begin{cases} 1, \text{ если } V_{i,2} = V_{2i-1}, \\ 0, \text{ если } V_{i,2} = V_{2i}. \end{cases}$$

Тогда мы имеем:

$$\begin{aligned}
F(\pi') &= F(\pi) + \sum_{i=1}^{n} x_i \left[(w_{2i-1} - w_{2i}) \cdot T_{V_{2i}}(\pi) - (p_{2i-1} - p_{2i}) \left(\sum_{j=1}^{i-1} w_{V_{j,2}} \right) \right] \\
&= F(\pi) + \sum_{i=1}^{n} x_i \left[b_i \cdot p_{2i} - b_i \cdot \left(\sum_{j=1}^{i-1} w_{V_{j,2}} \right) \right] \\
&= F(\pi) + \sum_{i=1}^{n} x_i \left[b_i \cdot p_{2i} - b_i \cdot \left(\sum_{j=1}^{i-1} w_{2j} + \sum_{j=1}^{i-1} x_j b_j \right) \right] \\
&= F(\pi) + \sum_{i=1}^{n} x_i b_i \left[p_{2i} - \frac{1}{2} \sum_{j=i+1}^{n} x_j b_j - \sum_{j=1}^{i-1} w_{2j} - \frac{1}{2} \sum_{j=1}^{i-1} x_j b_j \right] \\
&= F(\pi) + \frac{1}{2} \sum_{i=1}^{n} x_i b_i \left(\sum_{j \in N \setminus \{i\}} b_j - \sum_{j \in N \setminus \{i\}} x_j b_j \right) \\
&= F(\pi) + \frac{1}{2} \sum_{i=1}^{n} \sum_{j \in N \setminus \{i\}} x_i \cdot b_i \cdot b_j \cdot (1 - x_j).
\end{aligned}$$

В случае, когда существует подмножество $N' \subset N$ для примера ЗАДАЧИ РАЗБИЕНИЯ такое, что $\sum\limits_{i \in N'} b_i = \frac{1}{2} \sum\limits_{i \in N} b_i$, тогда при оптимальном каноническом расписании π^*, мы имеем

$$F(\pi^*) = F(\pi) + \frac{1}{2} A^2,$$

где $x_i = 1$, если $i \in N'$, и $x_i = 0$, если $i \in N \setminus N'$, т.к.

$$F(\pi^*) = F(\pi) + \frac{1}{2} \sum_{i \in N'} \sum_{j \in N \setminus N'} b_i \cdot b_j = F(\pi) + \frac{1}{2} A \cdot A.$$

Если ответ в примере ЗАДАЧИ РАЗБИЕНИЯ “НЕТ”, тогда

$$\begin{aligned} F(\pi^*) &= F(\pi) + \frac{1}{2} \sum_{i=1}^{n} \sum_{j \in N \setminus \{i\}} x_i \cdot b_i \cdot b_j \cdot (1 - x_j) \\ &= F(\pi) + \frac{1}{2}(A - y)(A + y) \\ &= F(\pi) + \frac{1}{2} A^2 - \frac{1}{2} y^2, \end{aligned}$$

где $y > 0$.

Максимальное значение целевой функции задачи $1(nd)\|\max \sum w_j T_j$ достигается тогда и только тогда, когда ответ в примере ЗАДАЧИ РАЗБИЕНИЯ “ДА”. □

Для задачи $1(nd)||\max \sum w_j T_j$ существует псевдополиномиальный алгоритм решения, поэтому задача NP-трудна в обычном смысле (не в сильном смысле).

9.11.2 Псевдополиномиальный алгоритм решения задачи $1(nd)||\max \sum T_j$

Алгоритм 9.5 основан на лемме 9.10, на том факте, что существует оптимальное расписание вида $\pi = (G, H)$, где все требования $j \in H$ запаздывают, а все требования $i \in G$ не запаздывают. Все требования из множества G обслуживаются в порядке неубывания продолжительности обслуживания (SPT), а все требования из множества H обслуживаются в порядке невозрастания продолжительности обслуживания (LPT).

Теорема 9.11 *Алгоритм 9.5 строит оптимальное расписание для задачи* $1(nd)||\max \sum T_j$ *за* $O(n \sum p_j)$ *операций.*

Доказательство теоремы аналогично доказательству теоремы 9.5. □

9.11.3 Графический алгоритм решения задачи $1(nd)||\max\sum T_j$

В этом разделе представлен Графический Алгоритм (GrA) решения задачи $1(nd)||\max\sum T_j$, основанный на уже известной нам идее модификации алгоритма динамического программирования. GrA представляет собой модификацию алгоритма 9.5, в которой функция $f_l(t)$ определена для любого значения $t \in (-\infty, +\infty)$ (не только для целых t), но мы вычисляем ее значения только в *точках излома*. Как будет показано далее, количество таких точек полиномиально. В этом параграфе помимо самого алгоритма GrA проиллюстрирована работа алгоритма на примере.

Алгоритм 9.5 Алгоритм решения задачи $1(nd)||\max\sum T_j$.

Перенумеруем требования согласно правилу: $p_1 \geq p_2 \geq \ldots \geq p_n$. Если $p_i = p_{i+1}$, тогда $d_i \geq d_{i+1}$;
for $t := 0$ to $\sum_{i=2}^{n} p_i$ **do**
  $\pi_1(t) := (1)$;
  $f_1(t) := \max\{0, p_1 + t - d_1\}$;
end for
for $l := 2$ to n **do**
  for $t := 0$ to $\sum_{i=l+1}^{n} p_i$ **do**
    $\pi^1 := (j, \pi_{j-1}(t + p_j))$, $\pi^2 := (\pi_{j-1}(t), j)$;
    $\Phi^1(t) := \max\{0, p_l + t - d_l\} + f_{l-1}(t + p_l)$;
    $\Phi^2(t) := f_{l-1}(t) + \max\{0, \sum_{j=1}^{l} p_j + t - d_l\}$;
    if $\Phi^1(t) < \Phi^2(t)$ **then**
      $f_l(t) := \Phi^2(t)$;
      $\pi_j(t) := \pi^2$;
    else
      $f_l(t) := \Phi^1(t)$;
      $\pi_j(t) := \pi^1$;
    end if
  end for
end for

$\pi_n(0)$ – оптимальное расписание, а $f_n(0)$ – соответствущее оптмальное значение целевой функции (максимальное суммарное запаздывание).

На каждом шаге алгоритма GrA, мы сохраняем функцию $f_l(t)$ в табличном виде, как представлено в таб. 9.5.

Записи в таблице означают следующее. Для каждого значения $t \in$ $\in (t_{k-1}, t_k]$, оптимальным будет частичное расписание π_k^l, в котором u_k запаздывающих требований, и которому соответствует значение целевой функции

Таблица 9.5: Функция $f_l(t)$

t	$(-\infty, t_1]$	$(t_1, t_2]$	...	$(t_w, +\infty)$
b	$b_1 = 0$	b_2	...	b_{w+1}
кол-во запазд.треб-ий u	$u_1 = 0$	u_2	...	u_{w+1}
частич.оптим.расписание	π_1^l	π_2^l	...	π_{w+1}^l

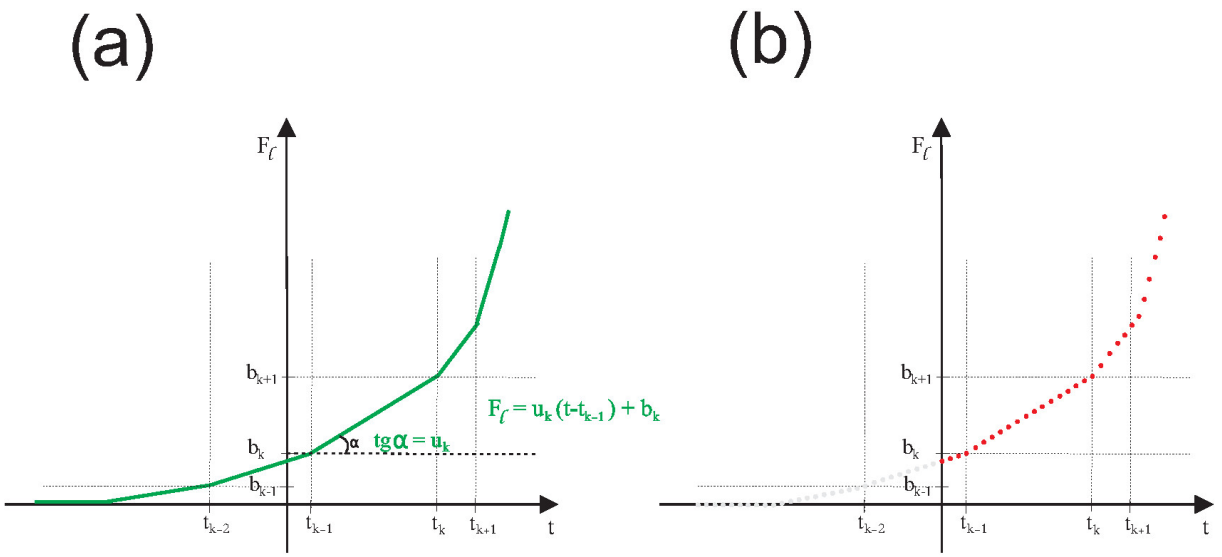

Рис. 9.3: Функция $f_l(t)$ в алгоритме 9.5 и в алгоритме GrA

$f_l(t) = u_k \cdot (t - t_{k-1}) + b_k$ (см. рис. 9.3). Функция $f_l(t)$ определена не только для целых t, но и для вещественных t.

Для упрощения описания алгоритма GrA мы рассматриваем всю ось t, т.е. $t \in (-\infty, +\infty)$. В теореме 9.12 показано, что эта таблица соответствует непрерывной, кусочно-линейной выпуклой функции $f_l(t)$. Точки $t_1, t_2, \ldots, t_w$ называются *точками излома*. Только в них происходит изменение (увеличение) количества запаздывающих требований с u_{k-1} на u_k (это означает, что меняется наклон кусочно-линейной функции). Заметим, что точки t_k могут быть нецелочисленными. Для описания каждого линейного сегмента функции, мы храним его наклон u_k и значение функции b_k в точке $t = t_{k-1}$.

В GrA функция $f_l(t)$ соответствует тем же рекурсивным уравнениям Беллмана, что и функция $f_l(t)$ в алгоритме 9.5, то есть для каждого $t \in$ $\in Z \bigcap [0, \sum_{j=2}^{n} p_j]$, $f_l(t)$ имеет тоже значение, что и в алгоритме 9.5 (см. рис.9.3), но теперь функция определена на всем интервале $t \in (-\infty, +\infty)$. В результате, "состояния" t, соответствующие одному и тому же оптимальному частичному расписанию, сгруппированы в интервалы. На рис. 9.3 (a), показана функция $f_l(t)$ из GrA, а на рис. 9.3 (b) – функция $f_l(t)$ из алгоритма 9.5.

9.11.4 Графический алгоритм GrA

Шаг 1. Перенумеруем требования согласно правилу:
$p_1 \geq p_2 \geq \ldots \geq p_n$. Если $p_i = p_{i+1}$, тогда $d_i \geq d_{i+1}$.

Шаг 2. $l := 1$, $\pi_1^1(t) := (1)$, функция $f_1(t) := \max\{0, p_1 + t - d_1\}$ определена для всех t. Функция $f_1(t)$ задается таблицей 9.6.

Таблица 9.6: Функция $f_1(t)$

t	$(-\infty, d_1 - p_1]$	$(d_1 - p_1, +\infty)$
b	0	0
u	0	1
частич.оптим.расписание	(1)	(1)

Шаг 3. Пусть $l > 1$, а функция $f_{l-1}(t)$ вычислена (таблица 9.7). Из функции $f_{l-1}(t)$ построим функцию $f_l(t)$. Промежуточные функции $\Phi^1(t)$ и $\Phi^2(t)$ также хранятся в виде таблице 9.5.

Таблица 9.7: Функция $f_{l-1}(t)$

t	$(-\infty, t_1]$	$(t_1, t_2]$	$\ldots$	$(t_w, +\infty)$
b	$b_1 = 0$	b_2	$\ldots$	b_{w+1}
u	$u_1 = 0$	u_2	$\ldots$	u_{w+1}
частич.оптим.расписание	π_1^{l-1}	π_2^{l-1}	$\ldots$	π_{w+1}^{l-1}

Шаг 3.1. Функция $\Phi^1(t)$ строится из функции $f_{l-1}(t)$ при помощи следующих операций. Мы сдвигаем график функции $f_{l-1}(t)$ влево на p_l единиц и в таблицу, соответствующую "сдвинутой" функции $f_{l-1}(t)$, добавляем новую колонку, соответствующую новой точке излома $t' = d_l - p_l$. Если $t_k - p_l < t' < t_{k+1} - p_l$, $k + 1 \leq w$, тогда в таблицу $\Phi^1(t)$ мы добавляем два интервала: $(t_k - p_l, t']$ и $(t', t_{k+1} - p_l]$. Далее мы увеличиваем все значения $u_{k+1}, u_{k+2}, \ldots, u_{w+1}$ на 1, т.е. количество запаздывающих требований (и наклон соответствующего линейного сегмента графика) возрастает. Соответствующее частичное расписание π^1 строится добавлением требования l в начало предыдущего частичного расписания. Таким образом получена таблица 9.8, соответствующая функции $\Phi^1(t)$.

Таблица 9.8: Функция $\Phi^1(t)$

t	$(-\infty,$ $t_1 - p_l]$	$(t_1 - p_l,$ $t_2 - p_l]$	$\ldots$	$(t_k - p_l, t']$	$(t',$ $t_{k+1} - p_l]$	$(t_{k+1} - p_l,$ $t_{k+2} - p_l]$	$\ldots$	$(t_w - p_l,$ $+\infty)$
b	$b_1 = 0$	b_2	$\ldots$	b_{k+1}	b'	b'_{k+2}	$\ldots$	b'_{w+1}
u	$u_1 = 0$	u_2	$\ldots$	u_{k+1}	$u_{k+1} + 1$	$u_{k+2} + 1$	$\ldots$	$u_{w+1} + 1$
частич. оптим. распи-сание π^1	(l, π_1^{l-1})	(l, π_2^{l-1})	$\ldots$	(l, π_{k+1}^{l-1})	(l, π_{k+1}^{l-1})	(l, π_{k+2}^{l-1})	$\ldots$	(l, π_{w+1})

Где $b' = b_{k+1} + (t' - (t_k - p_l)) \cdot u_{k+1}$, $b'_{k+2} = b' + ((t_{k+1} - p_l) - t') \cdot (u_{k+1} + 1)$, $\ldots$, $b'_{m+1} = b_m{}' + ((t_m - t_{m-1}) \cdot (u_m + 1)$

Шаг 3.2. Функция $\Phi^2(t)$ строится из функции $f_{l-1}(t)$ при помощи следующих операций. В таблицу, соответствующую функции $f_{l-1}(t)$, добавляем новую колонку, соответствующую новой точке излома $t'' = d_l - \sum_{i=1}^{l} p_i$. Если $t_h < t'' < t_{h+1}$, $h + 1 \leq w$, тогда в таблицу $\Phi^2(t)$ мы добавляем два интервала: $(t_h, t'']$ и $(t'', t_{h+1}]$. Далее мы увеличиваем все значения $u_{h+1}, u_{h+2}, \ldots, u_{w+1}$ на 1, т.е. количество запаздывающих требований (и наклон соответствующего линейного сегмента графика) возрастает. Соответствующее частичное расписание π^2 строится добавлением требования l в конец предыдущего частичного расписания. Таким образом получена таблица 9.9, соответствующая функции $\Phi^2(t)$.

Таблица 9.9: Функция $\Phi^2(t)$

t	$(-\infty, t_1]$	$(t_1, t_2]$	$\ldots$	$(t_h, t'']$	$(t'', t_{h+1}]$	$(t_{h+1}, t_{h+2}]$	$\ldots$	$(t_w, +\infty)$
b	$b_1 = 0$	b_2	$\ldots$	b_{h+1}	b''	b''_{h+2}	$\ldots$	b''_{w+1}
u	$u_1 = 0$	u_2	$\ldots$	u_{h+1}	$u_{h+1} + 1$	$u_{h+2} + 1$	$\ldots$	$u_{w+1} + 1$
частич. оптим. распи-сание π^2	(π_1^{l-1}, l)	(π_2^{l-1}, l)	$\ldots$	(π_{h+1}^{l-1}, l)	(π_{h+1}^{l-1}, l)	(π_{h+2}^{l-1}, l)	$\ldots$	(π_{w+1}^{l-1}, l)

Где $b'' = b_{h+1} + (t'' - t_h) \cdot u_{h+1}$, $b''_{h+2} = b'' + (t_{h+1} - t'') \cdot (u_{h+1} + 1)$, $\ldots$, $b''_{w+1} = b_w{}'' + (t_w - t_{w-1}) \cdot (u_w + 1)$

Шаг 3.3. Теперь мы строим таблицу, соответствующую функции

$$f_l(t) = \max\{\Phi^1(t), \Phi^2(t)\}.$$

В интервалах, образованных точками из двух таблиц 9.8 и 9.9 мы сравниваем значения функций $\Phi^1(t)$ и $\Phi^2(t)$ и ищем точки пересечения их графиков. Этот шаг требует порядка $O(w)$ операций.

Более подробно – конструируем список $t_1, t_2, \ldots, t_s,\ t_1 < t_2 < \ldots, t_s$, всех точек t из обоих таблиц $\Phi^1(t)$ и $\Phi^2(t)$, которые являются границами интервалов из этих таблиц. Далее рассматриваем каждый интервал $(t_i, t_{i+1}]$, где $i = 1, 2, \ldots, s-1$, и сравниваем функции $\Phi^1(t)$ и $\Phi^2(t)$ на этом интервале. Пусть интервал $(t_i, t_{i+1}]$ включен в интервал $(t_{x-1}, t_x]$ из таблицы. $\Phi^1(t)$ и в интервал $(t_{y-1}, t_y]$ из таблицы $\Phi^2(t)$. Тогда $\Phi^1(t) = t \cdot u_x + b_x + (t_i - t_{x-1}) \cdot u_x$ и $\Phi^2(t) = t \cdot u_y + b_y + (t_i - t_{y-1}) \cdot u_y$. Выбираем колонку из двух таблиц, соответствующую максимуму из двух функций и вставляем эту колонку в таблицу 9.10. Если на данном интервале существует точка пересечения t''' функций $\Phi^1(t)$ и $\Phi^2(t)$, то мы вставляем две колонки, соответствующие интервалам $(t_i, t''']$ и $(t''', t_{i+1}]$.

Таблица 9.10: Функция $f_l(t)$

t	$(-\infty, t_1]$	$(t_1, t_2]$	$\ldots$	$(t_{k-1}, t_k]$	$(t_k, t_{k+1}]$	$\ldots$	$(t_w, +\infty)$
b	$b_1 = 0$	b_2	$\ldots$	b_k	b_{k+1}	$\ldots$	b_{w+1}
u	$u_1 = 0$	u_2	$\ldots$	u_k	u_{k+1}	$\ldots$	u_{w+1}
частич. оптим. расписание	π_1^l	π_2^l	$\ldots$	π_k^l	π_{k+1}^l	$\ldots$	π_{w+1}^l

Если в итоговой таблице 9.10, соответствующей функции $f_l(t)$, встречается ситуация как в таб. 9.11, мы удаляем колонку $(t_k, t_{k+1}]$ и объединяем оба интервала, т.е. принимаем $t_k := t_{k+1}$.

Таблица 9.11: Удаление колонки

t	$\ldots$	$(t_{k-1}, t_k]$	$(t_k, t_{k+1}]$	$\ldots$
b	$\ldots$	$\ldots$	$\ldots$	$\ldots$
u	$\ldots$	u_k	$u_{k+1} = u_k$	$\ldots$
частич.оптим.расписание π_l	$\ldots$	$\ldots$	$\ldots$	$\ldots$

В теореме 9.12 доказано, что для каждого l, число колонок в таблице 9.10 не больше $l + 1 \leq n + 1$.

Шаг 3.4. Если $l = n$, тогда переходим к шагу 4 иначе $l := l + 1$ и переходим к шагу 3.

Шаг 4. В таблице $f_n(t)$, мы находим колонку $(t_k, t_{k+1}]$, где $t_k < 0 \leq$ $\leq t_{k+1}$. Тогда мы имеем оптимальное расписание $\pi^* = \pi_{k+1}$ и оптимальное значение целевой функции $F(\pi^*) = b_{k+1} + (0 - t_k) \cdot u_{k+1}$.

Очевидно, что GrA строит тоже решение, что и алгоритм 9.5. В то время, как в алгоритме 9.5 рассматриваются все целочисленные точки $t \in$ $\in [0, \sum_{j=2}^{n} p_j]$, в GrA рассматриваются только точки излома, определенные на всей оси t, такие, что все значения $f_l(t)$, $l \in \{1, 2, \ldots, n\}$, $t \in Z \cap [0, \sum_{j=l+1}^{n} p_j]$ соответствует тем же значениям, определенным в алгоритме 9.5. Поэтому алгоритм GrA строит оптимальное решение.

Теорема 9.12 *Трудоёмкость алгоритма GrA составляет $O(n^2)$ операций.*

Доказательство. Покажем, что все функции $f_l(t)$, $l = 1, 2, \ldots, n$, являются непрерывными кусочно-линейными **выпуклыми** функциями.

Очевидно, что функция $f_1(t)$ – непрерывная кусочно-линейная выпуклая функция с одной точкой излома. Согласно шагу 3 алгоритма обе функции $\Phi^1(t)$ и $\Phi^2(t)$ также являются непрерывными, кусочно-линейными и выпуклыми. Поэтому функция $f_2(t) = \max\{\Phi^1(t), \Phi^2(t)\}$ также непрерывная, кусочно-линейная и выпуклая.

Продолжая данные рассуждения для других l, получим, что все функции $f_l(t), l = 1, 2, \ldots, n$, обладают этим характеристиками, т.е. являются непрерывными, кусочно-линейными и **выпуклыми**.

Теперь предположим, что на шаге 3 алгоритма мы получили таблицу 9.10 для некоторой функции $f_l(t)$. Покажем, что выполняется неравенство $u_1 < u_2 < \ldots < u_k < u_{k+1} < \ldots < u_{w+1}$. Предположим, что $u_k > u_{k+1}$. Тогда для каждого $t \in (t_k, t_{k+1}]$, где t – время начала обслуживания требований из частичного расписания (π_k или π_{k+1}), имеем $F(\pi_k) > F(\pi_{k+1})$, т.к. выполняется $u_k > u_{k+1}$. То есть получили противоречие[4]. Напомним, что функция $f_l(t)$ является непрерывной кусочно-линейной выпуклой функцией, и $b_{k+1} = b_k + (t_k - t_{k-1}) \cdot u_k$.

Согласно алгоритму GrA, если $u_k = u_{k+1}$, тогда мы удаляем колонку, соответствующую точке излома u_{k+1} и объединяем оба интервала.

Мы имеем $u_1 < u_2 < \ldots < u_k < u_{k+1} < \ldots < u_{w+1}$, где u_i, $i = 1, 2, \ldots, w+1$, – количество запаздывающих требований. Тогда $w + 1 \leq l + 1 \leq n + 1$. Как следствие, мы имеем не более l точек излома, что значит, не более $l+1 \leq$ $\leq n + 1$ колонок для каждой функции $f_l(t)$, $l = 2, \ldots, n$.

[4]Заметим, что $u_1 < \ldots < u_w$, можно доказать иначе. Функции $\Phi^1(t)$ и $\Phi^2(t)$ выпуклы, а следовательно функция $f_l(t) = \max\{\Phi^1(t), \Phi^2(t)\}$ также выпукла. Значения u_k соответствуют $tg\alpha$, где α угол между осью t и линейным сегментом функции $f_l(t)$.

Поэтому трудоёмкость Шага 3 алгоритма составляет $O(n)$ операций для каждого $l = 2, \ldots, n$. Значит трудоёмкость алгоритма GrA составляет $O(n^2)$ операций. □

Стоит заметить, что уже для задачи $1(nd)||\max\sum w_jT_j$ аналогичный алгоритм имеет трудоёмкость $O(\min\{2^n, n \cdot \min\{d_{\max}, \sum w_j\}\})$ операций.

9.11.5 Иллюстрация работы Графического Алгоритма на примере

Рассмотрим пример задачи $1||\max\sum T_j$, входные данные которого представлены в таблице 9.12.

Таблица 9.12: Входные данные

j	1	2	3	4
p_j	30	22	12	5
d_j	32	35	38	4

Опишем таблицы, сохраняемые на каждой итерации l, алгоритма GrA.

Пусть $l = 1$. Сохраняем таблицу 9.13.

Таблица 9.13: Функция $f_1(t)$

t	$(-\infty, 2]$	$(2, +\infty)$
$f_1(t)$	0	0
u	0	1
π_1	(1)	(1)

График функции $f_1(t)$ представлен на рис. 9.5 (а).

Пусть $l = 2$. Получаем следующие результаты. Вычисляем новую точку излома $t' = d_2 - p_2 = 13$, и таб.9.14 для функции $\Phi^1(t)$.

Таблица 9.14: Функция $\Phi^1(t)$

t	$(-\infty, -20]$	$(-20, 13]$	$(13, +\infty)$
$\Phi^1(t)$	0	0	33
u	0	1	2
π^1	(2,1)	(2,1)	(2,1)

Вычисляем новую точку излома $t' = d_2 - (p_1 + p_2) = -17$, и таб. 9.15 для функции $\Phi^2(t)$.

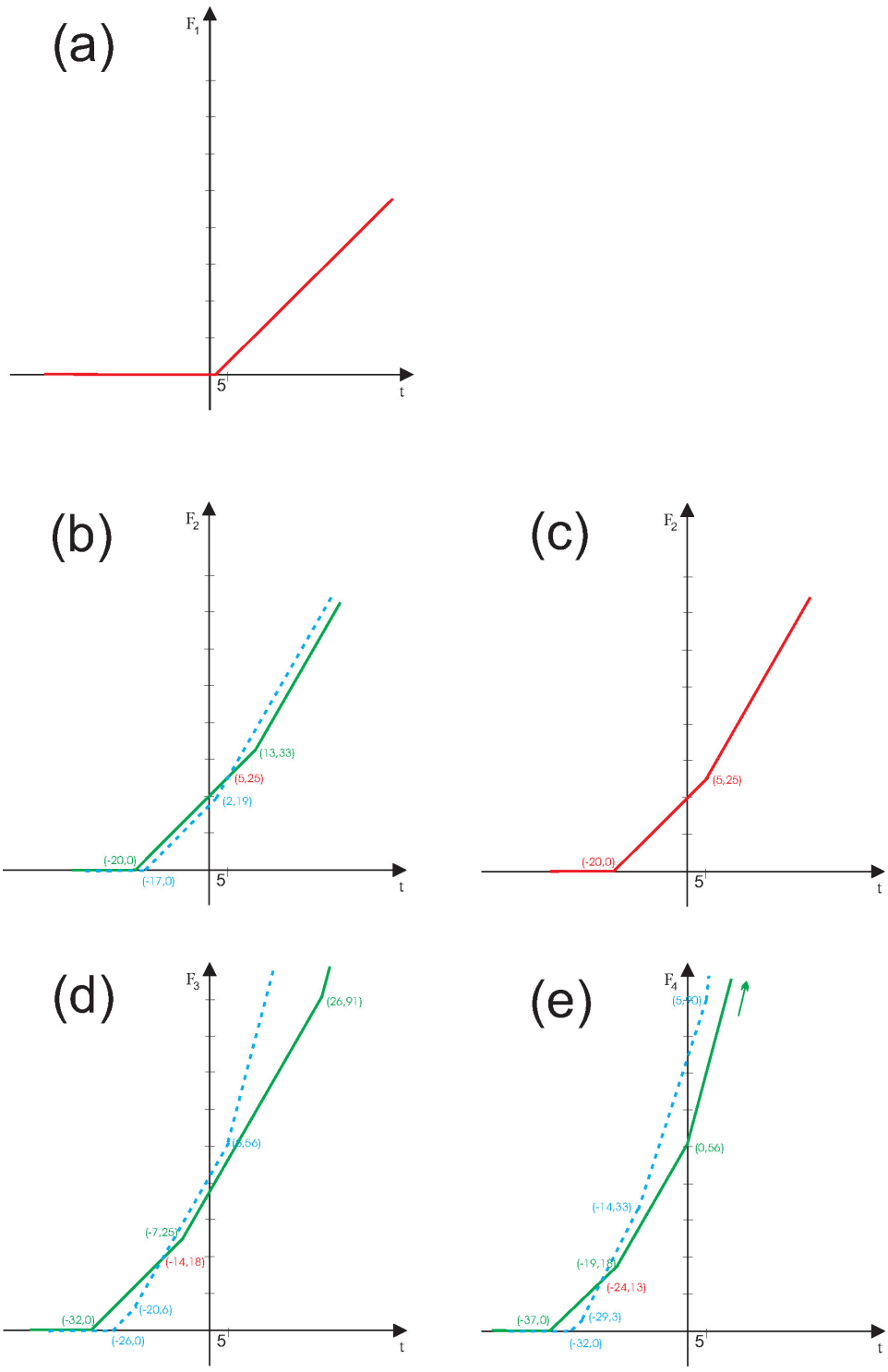

Рис. 9.4: Пример

Теперь проверим каждый из интервалов $(-\infty, -20]$, $(-20, -17]$, $(-17, 2]$, $(2, 13]$, $(13, +\infty)$, и проверим, не пересекаются ли функции $\Phi^1(t)$ и $\Phi^2(t)$ на этих интервалах. Для интервала $t \in (2, 13]$ из неравенства $\Phi^1(t) = (t + 20) \cdot 1 + 0 = = \Phi^2(t) = (t - 2) \cdot 2 + 19$ получаем точку пересечения $(5, 25)$. Комбинируя результаты из таблиц $\Phi^1(t)$ и $\Phi^2(t)$, получаем таблицу 9.16 функции $f_2(t)$.

Таблица 9.15: Функция $\Phi^2(t)$

t	$(-\infty,-17]$	$(-17,2]$	$(2,+\infty)$
$\Phi^2(t)$	0	0	19
u	0	1	2
π^2	(1,2)	(1,2)	(1,2)

Заметим, что для $t \leq 5$, частичное расписание (2,1), соответствующее функции $\Phi^1(t)$ является оптимальным, в то время, как для $t > 5$ оптимальным является частичное расписание $(1,2)$, соответствующее функции $\Phi^2(t)$.

Таблица 9.16: Функция $f_2(t)$

t	$(-\infty,-20]$	$(-20,5]$	$(5,+\infty)$
$f_2(t)$	0	0	25
u	0	1	2
π_2	(2,1)	(2,1)	(1,2)

График функции $f_1(t)$ представлен на рис. 9.5 (c). Он получен из графиков, представленных на рис. 9.5 (b). Аналогично рассмотрим оставшиеся итерации $l = 3, 4$.

Пусть $l = 3$. Получаем следующие результаты.

Вычисляем новую точку излома $t' = d_3 - p_3 = 26$, и таблицу 9.17 для функции $\Phi^1(t)$.

Таблица 9.17: Функция $\Phi^1(t)$ для $l = 3$

t	$(-\infty,-32]$	$(-32,-7]$	$(-7,26]$	$(26,+\infty)$
$\Phi^1(t)$	0	0	25	91
u	0	1	2	3
π^1	(3,2,1)	(3,2,1)	(3,1,2)	(3,1,2)

Вычисляем новую точку излома $t' = d_3 - (p_1 + p_2 + p_3) = -26$, и таб. 9.18 для функции $\Phi^2(t)$.

Таблица 9.18: Функция $\Phi^2(t)$ для $l = 3$

t	$(-\infty,-26]$	$(-26,-20]$	$(-20,5]$	$(5,+\infty)$
$\Phi^2(t)$	0	0	6	56
u	0	1	2	3
π^2	(2,1,3)	(2,1,3)	(2,1,3)	(1,2,3)

Точка пересечения двух графиков функций $\Phi^1(t)$ и $\Phi^2(t)$ – $(-14, 18)$. Функция $f_3(t)$ представлена в таблице 9.19. Ее график показан на рис. 9.5 (d).

Таблица 9.19: Функция $f_3(t)$

t	$(-\infty,-32]$	$(-32,-14]$	$(-14,5]$	$(5,+\infty)$
$f_3(t)$	0	0	18	56
u	0	1	2	3
π_3	(3,2,1)	(3,2,1)	(2,1,3)	(1,2,3)

Пусть $l = 4$. Получаем следующие результаты.

Вычисляем новую точку излома $t' = d_4 - p_4 = 35$ и таб.9.20 для функции $\Phi^1(t)$.

Таблица 9.20: Функция $\Phi^1(t)$ для $l = 4$

t	$(-\infty,-37]$	$(-37,-19]$	$(-19,0]$	$(0,35]$	$(35,+\infty)$
$\Phi^1(t)$	0	0	18	56	161
u	0	1	2	3	4
π^1	(4,3,2,1)	(4,3,2,1)	(4,2,1,3)	(4,1,2,3)	(4,1,2,3)

Вычисляем новую точку излома $t' = d_4 - (p_1 + p_2 + p_3 + p_4) = -29$ и таб. 9.21 для функции $\Phi^2(t)$.

Таблица 9.21: Функция $\Phi^2(t)$ для $l = 4$

t	$(-\infty,-32]$	$(-32,-29]$	$(-29,-14]$	$(-14,5]$	$(5,+\infty)$
$\Phi^2(t)$	0	0	3	33	90
u	0	1	2	3	4
π^2	(3,2,1,4)	(3,2,1,4)	(3,2,1,4)	(2,1,3,4)	(1,2,3,4)

Точка пересечения двух графиков функций $\Phi^1(t)$ и $\Phi^2(t)$ есть точка на плоскости $(-24, 13)$. Функция $f_4(t)$ представлена в таблице 9.22. Ее график показан на рис. 9.5 (e).

Таблица 9.22: Функция $f_4(t)$

t	$(-\infty,-37]$	$(-37,-24]$	$(-24,-14]$	$(-14,5]$	$(5,+\infty)$
$f_4(t)$	0	0	13	33	90
u	0	1	2	3	4
π_4	(4,3,2,1)	(4,3,2,1)	(3,2,1,4)	(2,1,3,4)	(1,2,3,4)

Мы получили оптимальное значение искомой целевой функции $f_4(0) = 33 + 14 \cdot 3 = 75$, которому соответствует оптимальное расписание $\pi_4(0) = (2, 1, 3, 4)$.

9.12 Задачи с одним невозобновимым ресурсом

В данном параграфе рассматриваются одноприборные задачи с невозобновимым ресурсом. Этот ресурс необходим для обслуживания требований, причем потребности в ресурсе у требований могут отличаться. В качестве такого ресурса могут выступать, например, деньги, топливо, прочие расходные невозобновимые материалы. Так как подобные задачи часто возникают в бюджетных организациях, для которых ресурсом являются деньги, эти задачи также называются *финансовыми*.

Постановка этих задач совпадает с классической постановкой одноприборных задач. Дополнительно заданы невозобновимый ресурс G (деньги, топливо и т.п.) и моменты времени поступления ресурса $\{t_0, t_1, \ldots, t_y\}$, $t_0 = 0$, $t_0 < t_1 < \ldots < t_y$. В каждый момент времени t_i, $i = 0, 1, \ldots, y$, поступает $G(t_i) \geq 0$ единиц ресурса. Для каждого требования $j \in N$, задано потребление ресурса $g_j \geq 0$, которое происходит в момент начала обслуживания. Выполняется равенство

$$\sum_{j=1}^{n} g_j = \sum_{i=0}^{y} G(t_i).$$

Обозначим через S_j время начала обслуживания требования j. Расписание $S = (S_{j_1}, S_{j_2}, \ldots, S_{j_n})$ описывает в том числе порядок обслуживания требований: $\pi = (j_1, j_2, \ldots, j_n)$. Расписание $S = (S_{j_1}, S_{j_2}, \ldots, S_{j_n})$ является допустимым, если выполняется, помимо классических условий, следующее неравенство:

$$\sum_{k=1}^{i} g_{j_k} \leq \sum_{\forall l:\ t_l \leq S_{j_i}} G(t_l), \quad i = 1, 2, \ldots, n.$$

Перестановку π также называют расписанием, т.к. по этой перестановке можно вычислить расписание $S = (S_{j_1}, S_{j_2}, \ldots, S_{j_n})$ за $O(n)$ операций.

Такие одноприборные задачи обозначают $1|NR, ...|...$, где NR обозначает присутствие в условии задачи невозобновимого ресурса (non-renewable).

В таблице 9.23 представлены некоторые сведения о трудоёмкости одноприборных задач с невозобновимым ресурсом.

В качестве иллюстрации приведем несложное доказательство NP-трудности некоторых из перечисленных задач.

Таблица 9.23: Трудоёмкость задач

Цел. функц.	**Частный случай**	**Трудоёмкость**
$C_{\max}$		NP-трудна в сильном смысле. Сведение ЗАДАЧИ 3-РАЗБИЕНИЯ
$\sum U_j$		NP-трудна в сильном смысле. Сведение ЗАДАЧИ 3-РАЗБИЕНИЯ
$\sum C_j$		NP-трудна в сильном смысле. Сведение $1\|r_j\|\sum C_j$
$L_{\max}$		NP-трудна в сильном смысле. Сведение ЗАДАЧИ 3-РАЗБИЕНИЯ
$\sum T_j$	$d_j = d$	NP-трудна в сильном смысле. Сведение ЗАДАЧИ 3-РАЗБИЕНИЯ
$\sum T_j$	$g_j = g$	NP-трудна (задача $1\|\|\sum T_j$ является частным случаем)
$\sum T_j$	$p_j = p$	NP-трудна. Сведение ЗАДАЧИ РАЗБИЕНИЯ.
$\sum T_j$	$d_j = d,\ g_j = g$	NP-трудна. Сведение ЗАДАЧИ РАЗБИЕНИЯ.
$\sum T_j$	$p_j = p$, $g_1 \le g_2 \le \ldots \le g_n$, $d_1 \le d_2 \le \ldots \le d_n$	полиномиально разрешима, оптимальное расписание: $\pi^* = (1, 2, \ldots, n)$

ЗАДАЧА 3-РАЗБИЕНИЯ. Задано множество $N = \{b_1, b_2, \ldots, b_n\}$ чисел, где $n = 3m$, $\sum\limits_{i=1}^{n} b_j = mB$ и $\frac{B}{4} \le b_j \le \frac{B}{2}$, $j = 1, 2, \ldots, n$. Необходимо ответить на вопрос, существует ли такое разбиение множества N на m подмножеств $N_1, N_2, \ldots, N_m$, каждое из которых содержит ровно три числа, и суммы чисел которых равны, т.е.

$$\sum_{b_j \in N_1} b_j = \sum_{b_j \in N_2} b_j = \ldots = \sum_{b_j \in N_m} b_j = B?$$

Известно, что ЗАДАЧА 3-РАЗБИЕНИЯ является NP-трудной в сильном смысле.

Теорема 9.13 *Задачи* $1|NR|C_{\max}$, $1|NR, d_j = d|\sum T_j$, $1|NR|\sum U_j$ *и* $1|NR|L_{\max}$ *являются NP-трудными в сильном смысле.*

Доказательство. Доказательство проведем сведением ЗАДАЧИ 3-РАЗБИЕНИЯ к частному случаю задач.

Рассмотрим задачу $1|NR|C_{\max}$. Дан произвольный пример ЗАДАЧИ 3-РАЗБИЕНИЯ, построим пример задачи $1|NR|C_{\max}$ следующим образом. В

пример зададим n требований с параметрами $g_j = p_j = b_j$, $j = 1, 2, \ldots, n$. Времена поступления ресурса определены как

$$\{t_0, t_1, \ldots, t_{m-1}\} = \{0, B, 2B, \ldots, (m-1)B\}$$

и $G(t_0) = G(t_1) = \ldots = G(t_{m-1}) = B$. Очевидно, что $C_{\max} = mB$ тогда и только тогда, когда ответ в примере ЗАДАЧИ 3-РАЗБИЕНИЯ "ДА". В этом случае в оптимальном расписании нет простоев прибора.

Для задач $1|NR, d_j = d|\sum T_j$, $1|NR|\sum U_j$ и $1|NR|L_{\max}$ дополнительно определим $d_j = d = mB$ для $j = 1, 2, \ldots, n$. Тогда $\sum T_j = 0$ выполняется тогда и только тогда, когда ответ в примере ЗАДАЧИ 3-РАЗБИЕНИЯ "ДА". Аналогично $\sum U_j = 0$ и $L_{\max} = 0$ выполняются тогда и только тогда, когда ответ в примере ЗАДАЧИ 3-РАЗБИЕНИЯ "ДА".

□

9.13 Интерполяционный подход к решению задач теории расписаний

Рассмотрим произвольную проблему теории расписаний $\alpha|\beta|F$: множество из n требований $J_1, \ldots, J_n$ с моментами поступления $r_1, \ldots, r_n$ продолжительностями обслуживания $p_1, \ldots, p_n$ и директивными сроками $d_1, \ldots, d_n$ должны быть обслужены на одном или нескольких приборах. Прерывания при обслуживании требований запрещены. Критерием задачи является минимизация некоторой регулярной функции от n аргументов $F(C_1, \ldots, C_n)$, где C_j — момент окончания обслуживания требования j. Предлагается подход нахождения приближенного значения целевой функции.

9.13.1 Аппроксимационный алгоритм

В данном параграфе для задачи $1||\sum T_j$ представлена ПОЛНАЯ ПОЛИНОМИАЛЬНАЯ АППРОКСИМАЦИОННАЯ СХЕМА (fully polynomial time approximation schema. FPTAS). Фактически, это приближенный алгоритм решения с настраиваемой погрешностью. То есть для любого числа $\varepsilon > 0$ с помощью алгоритма можно найти расписание π, такое, что $F(\pi) - F(\pi^*) \leq \leq \varepsilon F(\pi^*)$, где π^* – оптимальное расписание. Трудоемкость такого алгоритма полиномиально зависит от n и от значения $1/\varepsilon$. Поэтому алгоритм называется *полиномиальной аппроксимационной схемой*. Так как алгоритм работает

с любой заданной погрешностью $\varepsilon > 0$, схема называется *полной*. Существуют комбинаторные задачи, где построена *не полная* схема, при которой ε не может быть меньше некоторого числа $a > 0$.

Для описания схемы нам понадобится следующая лемма.

Лемма 9.12 *Пусть π_{EDD} – расписание, при котором требования упорядочены по неубыванию директивных сроков (т.е. по правилу EDD). Обозначим через $T_{max} = \max\{T_j(\pi_{EDD})\}$ – максимальное запаздывание при данном расписании. Тогда выполняется $T_{max} \le F(\pi^*) \le F(\pi_{EDD}) \le nT_{max}$, где π^* – оптимальное расписание.*

Элементарное доказательство этой леммы оставляем читателю в качестве упражнения.

Аппроксимационный алгоритм основан на точном алгоритме 9.3. Покажем, что трудоемкость точного алгоритма 9.3 не превосходит $O(n^5 T_{max})$. Обозначим через $T(N,t)$ – максимальное возможное суммарное запаздывание для множества требований N, обслуживание которых начинается с момента времени t. Несложно вычислить момент времени t^*, при котором $T(N,t) = 0,\ t < t^*$ и $T(N,t) > 0,\ t \ge t^*$, то есть момент времени, начиная с которого, в некоторых допустимых расписаниях суммарное запаздывание больше 0. Тогда очевидно, что в алгоритме 9.3 имеет смысл выполнять процедуру $ProcL$(N,t) только для точек t таких, что $t^* \le t < T_{EDD} < nT_{max}$. Тогда трудоемкость $O(n^4 \sum p_j)$ ограничена значением $O(n^5 T_{max})$.

Построим модифицированный пример, в котором все продолжительности обслуживания подверглись масштабированию, т.е. продолжительности обслуживания $q_j = \lfloor \frac{p_j}{K} \rfloor$, соответственно, директивные сроки $d'_j = \frac{p_j}{K},\ j = 1,2,\ldots,n$ (без округления). Предположим, что с помощью алгоритма 9.3 для данного модифицированного примера построили оптимальное расписание π. Обозначим через T'_A – суммарное запаздывание при этом расписании для примера с продолжительностями обслуживания $p'_j = Kq'_j$ и с директивными сроками, совпадающими с исходным примером (d_j). А через T_A – сумарное запаздывание для оригинального примера.

Очевидно, что $Kq_j \le p_j \le K(q_j + 1),\ j = 1,2,\ldots,n$. Используя это факт, покажем, что $T_A \le T'_A + K\frac{n(n+1)}{2}$. Обозначим через C'_j – моменты окончания обслуживания требований при расписании π для примера с продолжительностями обслуживания $p'_j = Kq'_j$. А через C_j – моменты окончания для исходного примера. Пусть $\pi = (j_1, j_2, j_3, \ldots, j_n)$. Тогда $C_{j_1} - C'_{j_1} \le K$, $C_{j_2} - C'_{j_2} \le 2K$, $C_{j_3} - C'_{j_3} \le 3K$ и т.д., а следовательно, $T_A - T'_A \le K(1 + 2 + \ldots + n) = K\frac{n(n+1)}{2}$.

Тогда мы получаем, что $T'_A \leq F(\pi^*) \leq T_A \leq T'_A + K\frac{n(n+1)}{2}$. Следовательно, $T_A - F(\pi^*) \leq K\frac{n(n+1)}{2}$. Зададим $K = \frac{2\varepsilon}{n(n+1)}T_{max}$. Тогда $T_A - F(\pi^*) \leq$ $\leq \varepsilon T_{max} \leq \varepsilon F(\pi^*)$, т.к. $T_{max} \leq F(\pi^*)$.

Таким образом, за время $O(n^5 T_{max}/K)$ эквивалентное $O(\frac{n^7}{\varepsilon})$ было найдено допустимое расписание π с относительной погрешностью не превосходящей ε.

Тогда аппроксимационный алгоритм можно описать следующим образом. Для выбранного значения ε вычисляется K и строится модифицированный пример с продолжительностями обслуживания $q_j = \lfloor \frac{p_j}{K} \rfloor$ и директивными сроками $d'_j = \frac{p_j}{K}$, $j = 1, 2, \ldots, n$. Для найденного примера с помощью точного алгоритма 9.3 строится оптимальное расписание π, которое является приближенным решением для исходного примера.

9.13.2 Аппроксимационная схема

Пусть задан пример $A = \{r_j, p_j, d_j\}$ исследуемой задачи теории расписаний. Для решения данного примера не хватает вычислительных мощностей известными точными методами... Идея предлагаемого подхода состоит в следующем: "проводим линию" в $3n$—мерном пространстве через точку (пример) A. Например, $\{(r_j, \gamma p_j, d_j) | j \in N\}$, где $\gamma \in [1, +\infty)$. При $\gamma = 1$, в данном случае, мы имеем исходный пример A. Затем на данной прямой задаем "разумные" граничные значения Γ_1 и Γ_2, $\Gamma_1 < \gamma < \Gamma_2$, $\Gamma_1 < 1 < \Gamma_2$. Особенностью большинства задач теории расписаний является то, что граничные значения интервала (примеры) могут быть вычислены за "разумное" время. Пусть H —"верхняя граница" вычислительных возможностей компьютера. Предлагается выбирать точки $\gamma_1, \ldots, \gamma_k$ как корни полинома Чебышёва на интервале $[\Gamma_1, \Gamma_2]$. Эти точки обладают хорошим свойством, они "прижимаются" к границам интервала. Для точек (примеров), соответствующих $\gamma_1, \ldots, \gamma_k$, для которых достаточно вычислительных возможностей компьютера (или кластера), находим значения целевой функции $F_1, \ldots, F_k$. Затем по точкам $(\gamma_1, F_1), \ldots, (\gamma_k, F_k)$ строим интерполяционный полином Чебышёва или Лагранжа. Погрешность целевой функции Δ для исходного примера (при $\gamma = 1$) не будет превышать величины $\Delta \leq \frac{\|(1-\gamma_1)\ldots(1-\gamma_k)\|}{(k+1)!}$. Схематично идея предлагаемого подхода показа на рис. 9.5. Как результат, мы не строим приближенное решение (расписание), а только оцениваем оптимальное значение целевой функции. Чем ближе мы выберем значения Γ_2 и Γ_1 и чем больше корней полинома Чебышёва, тем точнее (меньше) будет абсолютная погрешность целевой функции Δ.

Данный подход может быть применен для любой регулярной целевой функции.

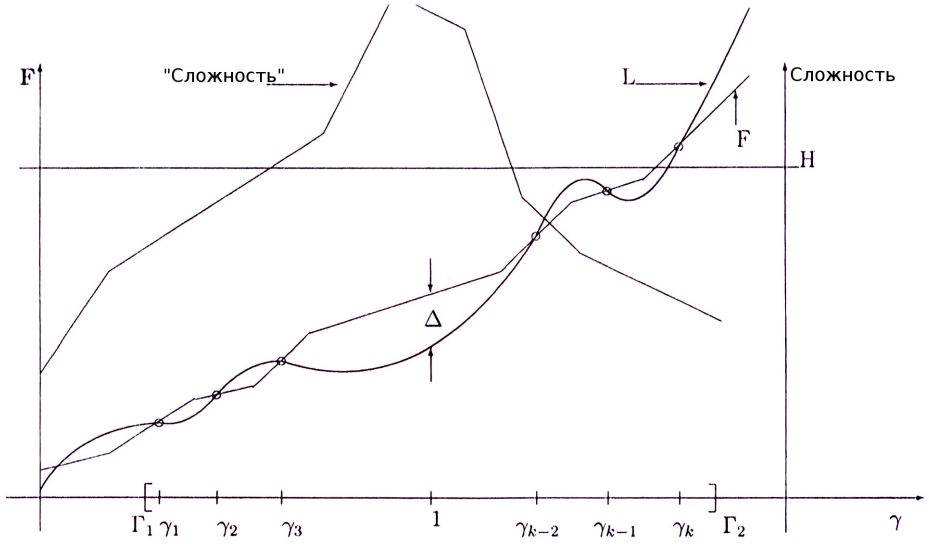

Рис. 9.5: Аппроксимационный полином: L — функции Лагранжа, F —целевая функция, H — "верхняя граница" вычислительных возможностей компьютера (сложность).

Заключение

Заинтересованному читателю предлагаем для размышлений следующие задачи, которые на наш взгляд являются интересными...

ЗАДАЧА 1. Каждый день мы выполняем некоторые работы, — решаем бытовые, личные проблемы, выполняем общественно значимые работы. Предположим, что в течении текущего дня человек из всей совокупности работ N, которые были доступны для обслуживания, выполнил подмножество работ $N_0 \subseteq N$. На множестве требований N заданы отношения предшествования в виде графа G. Расписание обслуживания этого множества требований π_0. Данное расписание человек составил из разумных (на его взгляд) соображений: он принимал во внимание всю совокупность проблем, стоящих перед ним, т.е. он выделил из N подмножество N_0 и оптимально его упорядочил во времени, построил расписание. Аналогично он поступал и в предыдущие дни, т.е. из текущей совокупности работ $N(t), t = 0, -1, -2, \ldots$, человек выбирал подмножество работ N_t и строил соответствующее оптимальное (на его взгляд) расписание π_t.

Таким образом, необходимо, во-первых, выбрать подмножество требований, а затем для выбранного подмножества построить оптимальное расписание. Термин "человек" заменим на "прибор". Будем предполагать, что подмножество требований и порядок обслуживания этих требований находится оптимальным образом.

Пусть $F(N_t)$ — целевая функция, которая "реализуется" на приборе. Необходимо по совокупности множества работ и расписаний $N(t), \pi_t, t = 0, -1, -2, \ldots, T$, построить целевую функцию $F(N_t)$.

Постройте регулярную целевую функцию от двух параметров $F(C_1, C_2)$, когда множества требований состояли всего из двух работ с параметрами $r_1, r_2; p_1, p_2$. Обе работы выполнялись каждый день, отношений предшествования между работами нет. Если целевую функцию не удается построить, то можно ли построить оптимальное расписание для двух работ зная $T + 1$ предшествующих оптимальных решений? То есть необходимо апроксимировать целевую функцию и на ее основе по исходным значениям параметров

построить расписание на "следующий день".

ЗАДАЧА 2. $1|r_j|\sum U_j$

Дано:
один прибор;
r_j — момент поступления требований j в систему;
p_j — продолжительность обслуживания требований $j, p_j > 0$;
d_j — директивный срок окончания обслуживания требований j;
w_j — вес (значимость) требований $j, j \in N$.
Запрещены:
— одновременное обслуживание более одного требования в каждый момент времени;
— прерывания при обслуживании.
Необходимо обслужить все множество требований $N = \{1, 2, \ldots, n\}$.
Целевая функция $\min\limits_{\pi} \sum\limits_{j \in N} U_j(\pi)$
при $r_j \in \{R^1, R^2\},\ w_j = 1, d_j \in \{D^1, D^2\},\ \forall j \in N$;
$C_j(\pi)$ — момент окончания обслуживания требования j при расписании π;
R^1, R^2, D^1, D^2 — некоторые константы $R^1 < R^2, D^1 < D^2$,
где $U_j(\pi) = 0$, если $C_j(\pi) \le d_j$, иначе $U_j(\pi) = 1$.
Необходимо построить алгоритм, трудоемкость которого не превышает $O(n^k)$ операций, нахождения оптимального расписания и обосновать его, где k — константа не зависящая от n.
Возможно данная задача является NP — трудной...

ЗАДАЧА 3. $1|r_j|L_{max}$

Дано
r_j — момент поступления j требований в систему;
p_j — продолжительность обслуживания требования $j, p_j > 0$;
d_j — директивный срок окончания обслуживания требования $j, j \in N$.
Запрещены:
— одновременное обслуживание более одного требования в каждый момент времени;
— прерывания при обслуживании.
Необходимо обслужить все множество требований $N = \{1, 2, \ldots, n\}$.
Целевая функция $\min\limits_{\pi} \max\limits_{j \in N}(C_j(\pi) - d_j)$
при $d_j = \alpha r_j + \beta p_j,\ j \in N$, где $C_j(\pi)$ —момент окончания обслуживания требования j при расписании π, α, β —некоторые константы.
Необходимо построить алгоритм, трудоемкость которого не превышает $O(n^3 \log n)$ операций, нахождения оптимального расписания и обосновать его.

Также необходимо показать, что данная оценка трудоемкости является нижней границей.

ЗАДАЧА 4. $1\|\sum w_jT_j$

Дано

r_j — момент поступления требований j в систему;

p_j — продолжительность обслуживания требований $j, p_j > 0$;

d_j — директивный срок окончания обслуживания требований j;

w_j — вес (значимость) требований $j, j \in N$.

Запрещены:

— одновременное обслуживание более одного требования в каждый момент времени;

— прерывания при обслуживании.

Необходимо обслужить все множество требований $N = \{1, 2, \ldots, n\}$.

Целевая функция $\min\limits_{\pi} \sum\limits_{j \in N} w_j max\{0, C_j(\pi) - d_j\}$

при $r_j = 0,\ d_j = \{D^1, D^2\},\ j \in N,$

где $C_j(\pi)$ — момент окончания обслуживания требования j при расписании $\pi,\ D^1,\ D^2$ — некоторые константы.

Необходимо построить алгоритм, трудоемкость которого не превышает $O(n^2)$ операций, нахождения оптимального расписания и обосновать его.

ЗАДАЧА 5. $1|r_j|\sum w_jC_j$

Дано

r_j — момент поступления требований j в систему;

p_j — продолжительность обслуживания требований $j, p_j > 0$;

w_j — вес (значимость) требований $j, j \in N$.

Запрещены:

— одновременное обслуживание более одного требования в каждый момент времени;

— прерывания при обслуживании.

Необходимо обслужить все множество требований $N = \{1, 2, \ldots, n\}$.

Целевая функция $\min\limits_{\pi} \sum\limits_{j \in N} w_jC_j(\pi),$

при $r_j \in \{R^1, R^2\},\ j \in N$, где $C_j(\pi)$ — момент окончания обслуживания требования j при расписании $\pi,\ R^1,\ R^2$ — некоторые константы.

Необходимо построить алгоритм, трудоемкость которого не превышает $O(n^2)$ операций, нахождения оптимального расписания и обосновать его.

ЗАДАЧА 6. $1|r_j, pmtn, p_j = p|\sum w_jC_j$

Дано

r_j — момент поступления требований j в систему;
p_j — продолжительность обслуживания требований $j, p_j = p$;
w_j — вес (значимость) требований $j, j \in N$.
Запрещены:
— одновременное обслуживание более одного требования в каждый момент времени;
Прерывания при обслуживании требований **разрешены**.
Необходимо обслужить все множество требований $N = \{1, 2, \ldots, n\}$.
Целевая функция $\min\limits_{\pi} \sum\limits_{j \in N} w_j C_j(\pi)$,
Возможно данная задача является NP — трудной...
Проанализировать случай, когда $r_j = j - 1;\ p_j = 2;\ w_{j+1} = w_j + 1,\ j = 1, 2, \ldots, n,\ w_{n+1} = +\infty$.

ЗАДАЧА 7. Рассмотрим однокритериальную задачу теории расписаний без искусственных простоев приборов.

Пусть π_t все множество допустимых расписаний и $F_1 < F_2 < \ldots < F_k$ все возможные различные значения целевой функции.
Гипотеза 1: Величина k (количество разных значений целевой функции) ограничено некоторым полиномом от размерности задачи.
Гипотеза 2: Примеры задач, у которых оптимальное значение целевой функции (F_1 — для задачи минимизации, F_k —для задачи максимизации) достигается на единственном расписании, решаются "проще".

Данные задачи отнюдь не покрывают всё множество открытых, нерешенных проблем...

Автор будет очень признателен заинтересованному читателю за присланные, интересные на его взгляд, нерешенные задачи.

Литература

[1] *Беллман Р.* Динамическое программирование //М.: ИЛ, 1960.– 400 с.

[2] *Белялов Т.К., Коннов И.В., Лазарев А.А., Перфилов С.Н., Халитов Ф.Р.* Программное обеспечение оптимального управления систем ГАП-ТВ.// Материалы научно-исследовательской конференции "Практика использования мини и микро ЭВМ в ГАП механической обработки" 1988, Казань. С.110–125.

[3] *Беркута О.Н., Лазарев А.А.* Алгоритмы сложности $O(n^3)$ решения задачи суммарного запаздывания.// Иссл. по прикл. мат. – Выпуск 22. – Казань, 1995. – С. 67–78.

[4] *Бурдюк В.Я., Шкурба В.В.* Теория расписаний. Задачи и методы решений // Кибернетика.– 1971. – № 1. – С. 89–102.

[5] *Бурков В.Н., Ловецкий С.Е.* Методы решения экстремальных комбинаторных задач (обзор) // Известия АН СССР, Техническая кибернетика. – 1968. – № 4. – С. 82–93.

[6] *Гафаров Е.Р., Лазарев А.А.* Доказательство $NP-$трудности одного частного случая задачи минимизации суммарного запаздывания// Известия РАН. Теория и системы управления.– 2006.– N 3.–
С. 120 – 128.

[7] *Гэри М., Джонсон Д.* Вычислительные машины и труднорешаемые задачи: Пер. с англ. // М.: Мир. – 1982. – 416 с.

[8] *Карп Р.М.* Сводимость комбинаторных проблем // В кн.: Кибернетический сборник. – М.: Мир. – 1972. – Вып. 12. – С. 16–38.

[9] *Каширских К.Н., Поттс К.Н., Севастьянов С.В.* Улучшенный алгоритм решения двухмашинной задачи flow shop с неодновременным поступлением работ// Дискретный анализ и исследование операций.– 1997.– Т. 4, N 1.– С. 13 – 32.

[10] *Кварацхелия А.Г., Лазарев А.А.* Метрики в задачах теории расписаний // Доклады Академии Наук, 2010. № т. 432 н. 6. С. 4

[11] *Кнут Д.* Искусство программирования для ЭВМ. т. 3: Сортировка и поиск.//М.: Мир. – 1973. – 348 с.

[12] *Ковалёв М.Я.* Интервальные ϵ – приближённые алгоритмы решения дискретных экстремальных задач // Дисс. канд. физ.-мат. наук. – Минск, 1986. – 110 с.

[13] *Конвей Р.В., Максвелл В.Л., Миллер Л.В.* Теория расписаний.// М.: Наука. Гл. ред. физ.-мат. лит., 1975.– 360 с. Англ. вариант книги: *Conway R.W., Maxwell W.L., Miller L.W.* Theory of Scheduling // Addison-Wesley, Reading, MA. – 1967.

[14] *Корбут А.А., Сигал И.Х., Финкельштейн Ю.Ю.* Методы ветвей и границ. Обзор теорий, алгоритмов, программ и приложений // Operations Forsch. Statist., Ser. Optimiz. -– 1977. -– V.8, N.2. -–
P. 253-–280.

[15] *Корбут А.А., Сигал И.Х., Финкельштейн Ю.Ю.* Гибридные методы в дискретном программировании // Изв. АН СССР. Техн. кибернет. — 1988. – № 1. — С. 65-–77.

[16] *Корбут А.А., Финкельштейн Ю.Ю.* Приближённые методы дискретного программирования // Изв. АН СССР. Техн. кибернет. — 1983. – № 1. — С. 165-–176.

[17] *Лазарев А.А.* Эффективный алгоритм для задачи минимизации максимальной задержки.// Теория управления и методы оптимизации.– Деп. в ВИНИТИ № 6633-83 С.14–21.

[18] *Лазарев А.А.* Алгоритмы теории расписаний, основанные на необходимых условиях оптимальности.// Исследования по прикладной математике. – Казань, 1984. – Выпуск 10. С. 102–110.

[19] *Лазарев А.А.* Двойственная проблема к задаче минимизации максимального штрафа.// Исследования по прикладной математике. – Казань, 1984. – Выпуск 10. С. 111–113.

[20] *Лазарев А.А.* Исследование задач теории расписаний с помощью преобразований.// Исследования по прикладной математике. – Казань, 1985. – Выпуск 12. С. 63–74.

[21] *Лазарев А.А.* Алгоритмы решения задачи минимизации суммарного запаздывания для одного прибора..// Теория управления и методы оптимизации.– Деп. в ВИНИТИ № 3795-85 С.21–30.

[22] *Лазарев А.А.* К решению задачи минимизации суммарного запаздывания.// VII конференция "Проблемы теоретической кибернетики 1985, Тезисы , часть I, Иркутск, С.114–115.

[23] *Лазарев А.А.* Об одном эффективном алгоритме решения задачи минимизации суммарного запаздывания.// X-ый симпозиум "Программное обеспечение решения задач оптимального планирования Нарва, 1988, Тезисы докладов. С.135–136.

[24] *Лазарев А.А.* Эффективные алгоритмы решения некоторых задач теории расписаний для одного прибора с директивными сроками обслуживания требований // Дис. канд. физ.-мат. наук. – Казань, 1989. – 108 с.

[25] *Лазарев А.А.* Алгоритмы декомпозиционного типа решения задачи минимизации суммарного запаздывания // Исследования по прикладной математике. – Казань: Изд-во Казан. гос. ун-та, 1990. – Вып. 17. – С. 71–78.

[26] *Лазарев А.А.* Декомпозиционные алгоритмы решения задачи минимизации суммарного запаздывания.// Исследования по прикладной математике, Вып. 21, Казань, 1994. С. 105–110.

[27] *Лазарев А.А.* Минимизация оценки абсолютной погрешности для задач $1||T_j$ и $1||V_j$.// Исследования по прикладной математике, Вып. 22, Казань, 1995. С. 56–62.

[28] *Лазарев А.А.* Парето-оптимальное множество $NP-$трудной задачи минимизации максимального временно́го смещения// Известия РАН. Теория и системы управления.– 2006.– N 6.– С. 103 – 110.

[29] *Лазарев А.А.* Оценка абсолютной погрешности задач теории расписаний с критерием минимизации максимального временно́го смещения.// Доклады Академии Наук. – 2007. – Том 415, № 4. – С. 446–449. (Англ. вариант статьи: *A. A. Lazarev* Estimation of Absolute Error in Scheduling Problems of Minimizing the Maximum Lateness// Doklady Mathematics. – 2007. – Vol. 76, No. 1. – P. 572—574).

[30] *Лазарев А.А.* Решение NP-трудной задачи теории расписаний минимизации суммарного запаздывания.//Журнал вычислительной математики и математической физики. – 2007. – Том 47, № 6. –
С. 1087–1099.

[31] *Лазарев А.А.* Графический подход к решению задач комбинаторной оптимизации.// Автоматика и Телемеханика. – 2007. – № 4. – С. 13–23.

[32] *Лазарев А.А., Баранов А.В.* Лазарев А.А., Баранов А.В. Graphical Approach for Combinatorial Problems / Proceedings of the 2nd International conference «Optimization and Applications» (Optima-2011, Petrovac, Montenegro). М.: Учреждение Российской академии наук Вычислительный центр им. А.А.Дородницына РАН, 2011. С. 149-152.

[33] *Лазарев А.А., Гафаров Е.Р.* Теория расписаний. Минимизация суммарного запаздывания для одного прибора. //М.: Научное издание. Вычислительный центр им. А.А. Дороницына РАН, 2006. – 134 с.

[34] *Лазарев А.А., Гафаров Е.Р.* Теория расписаний. Исследование задач с отношениями предшествования и ресурсными ограничениями.//М.: Научное издание. Вычислительный центр им. А.А. Дородницына РАН – 2007. – 80 с.

[35] *Лазарев А.А., Кварацхелия А.Г.* Стохастическое исследование методов ветвей и границ для NP-трудной задачи теории расписаний минимизации суммарного запаздывания для одного приборе $1 \mid\mid \sum T_j$ // Материалы VII Международного семинара «Дискретная математика и ее приложения» (Москва, 29 янв.–2 февр. 2001 г.). Часть III. – М.: Изд-во центра прикладных исследований при мех.-мат. факультете МГУ, 2001. – С. 379–381.

[36] *Лазарев А.А., Кварацхелия А.Г.* Исследование проблемы теории расписаний $1 \mid\mid \sum T_j$. // Проблемы теоретической кибернетики. Тезисы докладов XIII Межд. конференции «Проблемы теоретической кибернетики» (Казань, 27–31 мая 2002 г.). Часть II. – М: Изд-во центра прикладных исследований при мех.-мат. факультете МГУ, 2002. – С. 107.

[37] *Лазарев А.А., Кварацхелия А.Г.* К исследованию проблемы теории расписаний $1 \mid\mid \sum T_j$. // Материалы Российской конференции «Дискретный анализ и исследование операций», 24–28 июня, 2002 г. – Новосибирск, 2002. – С. 218.

[38] *Лазарев А.А., Кварацхелия А.Г.* Теоретическое и экспериментальное исследование NP-трудной проблемы теории расписаний минимизации суммарного запаздывания на одном приборе. // Исследования по прикладной математике и информатике. – Казань: Изд-во Казан. гос. ун-та, 2003. – Вып. 24. – С. 90–106.

[39] *Лазарев А.А., Кварацхелия А.Г.* Исследование NP-трудной проблемы теории расписаний минимизации суммарного запаздывания на одном

приборе, // Исследования по прикладной математике и информатике. – Казань: Изд-во Казанского гос. ун-та, 2003. – Вып. 24. – С. 90–106.

[40] *Лазарев А.А., Кварацхелия А.Г.* Исследование проблемы теории расписаний $1 \mid\mid \sum T_j$. // Материалы Российской конференции DAOR'02 «Дискретный анализ и исследование операций» (Новосибирск, 24—28 июня, 2002). – Новосибирск: Изд-во Ин-та математики, 2004. – С. 218.

[41] *Лазарев А.А., Кварацхелия А.Г.* Алгоритм $O(n^2 \sum p_j)$ решения NP - трудной проблемы теории расписаний $1 \mid\mid \sum T_j$. // Материалы VIII Межд. семинара «Дискретная математика и ее приложения» (2–6 февраля 2004 г.). – М: Изд. механико-математического факультета МГУ, 2004. – С. 211–213.

[42] *Лазарев А.А., Кварацхелия А.Г.* Алгоритм $O(n^2 \sum p_j)$ решения NP-трудной проблемы теории расписаний $1 \mid\mid \sum T_j$. // Материалы VIII Международного семинара «Дискретная математика и её приложения» (2–6 февраля 2004 г.). – Изд-во механико-математического факультета МГУ, 2004. – С. 211-213.

[43] *Лазарев А.А., Кварацхелия А.Г.* Алгоритм решения проблем $1 \mid\mid \sum T_j$ и четно-нечетного разбиения. // Труды VI Международной конференции «Дискретные модели в теории управляющих систем» (7–11 декабря 2004 г.). М.: Изд-во факультета вычислительной математики и кибернетики МГУ, 2004 – С. 184–187.

[44] *Лазарев А.А., Кварацхелия А.Г., Гафаров Е.Р.* Алгоритмы решения NP-трудной проблемы минимизации суммарного запаздывания для одного прибора. // Доклады Академии Наук, 2007. – Том 412, № 6. – С. 739–742. (Англ. вариант статьи: *A. A. Lazarev, A. G. Kvaratskheliya, and E. R. Gafarov.* Algorithms for Solving the NP-Hard Problem of Minimizing Total Tardiness for a Single Machine // Doklady Mathematics. – 2007. – Vol. 75, No. 1. – P. 130—133).

[45] *Лазарев А.А., Садыков Р.Р.* Эффективность полиномиального алгоритма $O(n^3 \log n)$ для решения NP-трудной проблемы минимизации максимального временно́го смещения $1 \mid r_j \mid L_{\max}$// Материалы VII Международного семинара "Дискретная математика и её приложения 29 января – 2 февраля.– М.: Изд-во центра прикладных исследований при мех.-мат. факультете МГУ, 2001.– С. 381 – 383.

[46] *Лазарев А.А., Садыков Р.Р.* К исследованию проблемы теории расписаний $1 \mid r_j \mid L_{\max}$// Материалы российской конференции "Дискретный

анализ и исследование операций 24 июня – 28 июня.– Новосибирск: Изд-во Ин-та математики, 2002.– С. 219.

[47] *Лазарев А.А., Садыков Р.Р.* Схема приближённого решения проблемы $1 \mid r_j \mid L_{\max}$// Материалы российской конференции "Дискретный анализ и исследование операций 28 июня – 2 июля.– Новосибирск: Изд-во Ин-та математики, 2004.– С. 173.

[48] *Лазарев А.А., Садыков Р.Р.* Теория расписаний. Минимизация максимального временно́го смещения и суммарного взвешенного числа запаздывающих требований.//М.: Научное издание. Вычислительный центр им. А.А. Дородницына РАН – 2007. – 180 с.

[49] *Лазарев А.А., Садыков Р.Р., Севастьянов С.В.* Схема приближённого решения проблемы $1 \mid r_j \mid L_{\max}$// Дискретный анализ и исследование операций.– 2006.– Сер. 2.– Т. 13, N 1.– С. 57 – 76.

[50] *Лазарев А.А., Сираев Р.Р.* К решению задачи календарного планирования методом ветвей и границ//Материалы 3-ей международной конференции "Математические методы и компьютеры в экономике май, 26-27, 1998, Часть 1, Пенза, С. 27–28.

[51] *Лазарев А.А., Сираев Р.Р.* Решение задач календарного планирования методом ветвей и границ.//Материалы 11-ой Байкальской международной школы-семинара "Методы оптимизации и их применение Июль, 5-12, 1998, Секция 1: Математическое программирование., С. 159–163.

[52] *Лазарев А.А., Сираев Р.Р.* Системы обработки экономической информации. Часть I. Кредитование в банке.// Казань: Издательство Казанского математического общества, 1998. – 285 С.

[53] *Лазарев А.А., Хабибуллин Р.Ф.* Эффективные алгоритмы для некоторых задач теории расписаний.// VII конференция "Проблемы теоретической кибернетики 1985, Тезисы , часть I, Иркутск, С.115–116.

[54] *Лазарев А.А., Шульгина О.Н.* Псевдополиномиальный алгоритм решения NP-трудной задачи минимизации максимального временно́го смещения.//Материалы 11-ой Байкальской международной школы-семинара "Методы оптимизации и их применение Июль, 5-12, 1998, Секция 1: Математическое программирование., С. 163–167.

[55] *Лазарев А.А., Шульгина О.Н.* Задача минимизации максимального временно́го смещения для одного прибора: свойства, процедуры, алгоритмы// 9 – я Всероссийская конференция "Математическое программирование и приложения 22 – 26 февраля.– Екатеринбург, 1999.– С. 183 – 184.

[56] *Лазарев А.А., Шульгина О.Н.* К решению задачи планирования производственно – экономической деятельности предприятия// Иссл. по прикл. мат.– Выпуск 21.– Казань, 1999.– С. 155 – 169.

[57] *Лазарев А.А., Шульгина О.Н.* Полиномиально разрешимые частные случаи задачи минимизации максимального временного смещения// Ред. Журн. "Изв. Вузов. Математика".- Казан. ун-т.– Казань, 2000.– 11 с.– Деп в ВИНИТИ 28.11.00, N 3019-В00.

[58] Современное состояние теории исследования операций// Под ред. Н.Н.Моисеева.– М.: Наука, 1979.– 464 с.

[59] *Пападимитриу Х., Стайглиц К.* Комбинаторная оптимизация. Алгоритмы и сложность, М.: Мир, 1985, – 512 с.

[60] *Посыпкин М.А., Сигал И.Х., Галимьянова Н.Н.* Алгоритмы параллельных вычислений для решения некоторых классов задач дискретной оптимизации. // М.: Вычислительный центр им. А.А. Дороницына РАН, 2005. – 44 с.

[61] *Севастьянов С.В.* Геометрические методы и эффективные алгоритмы в теории расписаний// Дис. док. физ.-мат. наук.– Новосибирск: 2000.– 280 с.

[62] *Сигал И.Х., Иванова А.П.* Введение в прикладное дискретное программирование// М.: Физматлит, 2002.– 240 с.

[63] *Танаев В.С., Шкурба В.В.* Введение в теорию расписаний.// М.: Наука. Гл. ред. физ.-мат. лит., 1975.– 256 с.

[64] *Танаев В.С., Гордон В.С., Шафранский Я.М.* Теория расписаний. Одностадийные системы// М.: Наука. Гл. ред. физ.-мат. лит., 1989.– 384 с.

[65] *Танаев В.С., Сотсков Ю.Н., Струсевич В.А.* Теория расписаний. Многостадийные системы// М.: Наука, Гл. ред. физ.-мат. лит., 1989.– 328 с.

[66] *Танаев В.С., Ковалёв М.Я., Шафранский Я.М.* Теория расписаний. Групповые технологии. // Минск: Институт технической кибернетики НАН Беларуси, 1998. – 290 с.

[67] *Финкельштейн Ю.Ю.* Метод отсечения и ветвления для решения задач целочисленного линейного программирования// Изв. АН СССР. Техн. кибернет.– 1971.– N 4.– С. 34 – 38.

[68] *Финкельштейн Ю.Ю.* Приближённые методы и прикладные задачи дискретного программирования // М.: Наука, 1976.

[69] *Хачатуров В.Р., Веселовский В.Е., Злотов А.В. и др.* Комбинаторные методы и алгоритмы решения задач дискретной оптимизации большой размерности // М.: Наука. – 2000.

[70] *Adams J., Balas E., Zawack D.* The shifting bottleneck procedure for job shop scheduling// Manag. Sci.– 1988.– V. 34, N 3.– P. 391 – 401.

[71] *van den Akker J.M., Hoogeveen J.A., van de Velde S.L.* Parallel machine scheduling by column generation// Oper. Res.– 1999.– V. 47, N 6.– P. 862 – 872.

[72] *van den Akker J.M., Hurkens C.A.J., Savelsbergh M.W.P.* Time-indexed formulations for single-machine scheduling problems: column generation// INFORMS J. on Computing.– 2000.– V. 12, N 2.– P. 111 – 124.

[73] *Alidaee B., Gopalan S.* A note of the equivalence of two heuristics to minimize total tardiness // European Journal of Operation Research. – 1997. – V.96. – P. 514–517.

[74] *Alon N., Woeginger G.J., Yadid T.* Approximation schemes for scheduling on parallel machines// J. of Scheduling.– 1998.– V. 1.– P. 55 – 66.

[75] *Apt K.* Principles of Constraint Programming// Cambridge University Press, 2003.– 420 p.

[76] *Baker K.R.* Introduction to sequencing and scheduling. // Wiley, New York. – 1974.

[77] *Baker K.R., Bertrand W.M.* A dynamic priority rule for scheduling against due dates // Journal of Operation Management. – 1982. – V.3. – P. 37–42.

[78] *Baker K.R., Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G.* Preemtive scheduling of a single machine to minimize maximum cost subject to release dates and precedence constraints// Oper. Res.– 1983.– V. 31, N 2.– P. 381 – 386.

[79] *Baker K.R., Schrage L.* Finding am optimal sequence by dynamic programming: an extension to precedence-related tasks // Operations Research. – 1978. – V. 26. – P. 111–120.

[80] *Baker K.R., Su Z.-S.* Sequencing with due-dates and early start times to minimize maximum tardiness// Naval Res. Logist. Quart.– 1974.– V. 21.– P. 171 – 176.

[81] *Baptiste Ph.* An $O(n^4)$ algorithm for preemptive scheduling of a single machine to minimize the number of late jobs// Oper. Res. Letters.– 1999.– V. 24.– P. 175 – 180.

[82] *Baptiste Ph.* Scheduling equal-length jobs on identical parallel machines // Discret. Appl. Math. 2000. No. 103. 21–32 .

[83] *Baptiste Ph., Jouglet A., Le Pape C., Nuijten W.* A constraint-based approach to minimize the weighted number of late jobs on parallel machines// University of Technology of Compiégne, 2000.– Research Report N 2000/288.– 45 p.

[84] *Baptiste Ph., Le Pape C., Nuijten W.* Constraint-based scheduling: applying constraint programming to scheduling problems// Kluwer Academic Publishers, 2001.– 198 p.

[85] *Baptiste Ph., Peridy, L., Pinson E.* A branch and bound to minimize the number of late jobs on a single machine with release time constraints// European J. of Oper. Res.– 2003. V. 144, N 1. P. 1 – 11.

[86] *Baptiste, P. and Brucker, P. and Knust, S. and Timkovsky, V.* Ten notes on equal-execution-time scheduling // 4OR, Quarterly Journal of the Belgian, French and Italian Operations Research Societies, 2, 2004, P. 111–127.

[87] *Bellman R.* Mathematical aspects of scheduling theory // Journal of the Society of Industrial and Applaid Mathematics. – 1956. – Vol. 4. – P. 168–205.

[88] *Benders J.F.* Partitioning procedures for solving mixed-variables programming prolems// Numerische Mathematik.– 1962.– V. 4. – P. 238 – 252.

[89] *Blazewicz J., Ecker K., Pesch E., Schmidt G., Weglarz J.* Scheduling Computer and Manufactoring Processes. // Springer Berlin. – 1996.

[90] *Bockmayr A., Pisaruk N.* Detecting infeasibility and generating cuts for MIP using CP// Proceedings of the Fifth International Workshop CPAIOR'03, 8 – 10 may.– Montreal: 2003.– P. 16 – 30.

[91] *Bockmayr A., Kasper Th.* Branch-and-Infer: A unifying framework for integer and finite domain constraint programming// INFORMS J. Computing.– 1998.– V. 10.– P. 287 – 300.

[92] *Bratley P., Florian M., Pobillard P.* On sequencing with earliest starts and due dates with application to computing bounds for the $(n/m/G/F_{\max})$ problem// Naval Res. Logist. Quart.– 1973.– V. 20.– P. 57 – 67.

[93] *Brooks G.N., White C.R.* An algorithm for finding optimal or near – optimal solutions to the production scheduling problem// J. Ind. Eng.– 1965.– V. 16, N 1.– P. 34 – 40.

[94] *Brucker P., Lenstra J.K., Rinnoy Kan A.N.G.* Complexity of machine scheduling problems // Math. Cent. Afd. Math Beslisk. – Amsterdam, 1975. – BW 43. – 29 pp.

[95] *Brucker, P., Garey, M.R., Johnson, D.S.* (1977), Scheduling equal-length tasks under treelike precedence constraints to minimize maximum lateness // Math. Oper. Res. – 1977. –**2(3)**.– P. 275–284.

[96] *Brucker, P. and Hurink, J. and Kubiak, W.* Scheduling identical jobs with chain precedence constraints on two uniform machines. // Math. Methods Oper. Res., Mathematical Methods of Operations Research. – 1999. – 49. – 2. – P. 211-219.

[97] *Brucker, P. and Knust, S.* Complexity results for single-machine problems with positive finish-start time-lags, Computing, 1999, 63, P. 299–316.

[98] *Brucker P.* Scheduling Algorithms// Springer-Verlag, 2001.– 365 p.

[99] *Brucker P., Knust S.* Complex Scheduling// Springer, 2006.– 284 p.

[100] *Bruno, J. and Jones, III, J.W. and So, K.* Deterministic scheduling with pipelined processors, *IEEE Trans. Comput.*, 29, 1980, 4, – P. 308–316.

[101] *Carlier J.* The one-machine sequencing problem// European J. of Oper. Res.– 1982.– V. 11, N 1.– P. 42 – 47.

[102] *Carlier J., Pinson E.* A practical use of Jackson's preemptive schedulefor solving the job-shop problem// Annals of Oper. Res.– 1990.– V. 26.– P. 269 – 287.

[103] *Carlier J., Pinson E.* Adjustment of heads and tails for the job-shop problem// European J. of Oper. Res.– 1994.– V. 78.– P. 146 – 161.

[104] *Carroll D.C.* Heuristic sequencing of single and multiple components // PhD Thesis, Massachusets Institute of Technology. – 1965.

[105] *Chang S., Lu Q., Tang G., Yu W.* On decomposition of the total tardiness problem, // Oper. Res. Lett. – 1995. – V.17. – P. 221–229.

[106] *Chen Z-L., Powell W.B.* Solving parallel machine scheduling problems by column generation// INFORMS J. Computing.– 1999.– V. 11.– P. 78 – 94.

[107] *Chopard B., Tomassini M.* Problems, Algorithms, and Computational Complexity// An Introduction to Metaheuristics for Optimization. Natural Computing Series. Springer, Cham – 2018 – P. 1-14

[108] *Colombani Y., Heipcke T.* Mosel: an extensible environment for modelling and programming solutions// Proceedings of the Fourth International

Workshop CPAIOR'02, 25 – 27 march.– Le Croisic, France: 2002.– P. 277 – 290.

[109] *della Croce F., Grosso A., Paschos V.* Lower bounds on the approximation ratios of leading heuristics for the single-machine total tardiness problem // Journal of Scheduling. – 2004. – V.7. – P. 85–91.

[110] *Davida, G.I. and Linton, D.J.* A new algorithm for the scheduling of tree structured tasks // Proc. Conf. Inform. Sci. and Syst. – 1976. – Baltimore, MD. P. 543–548.

[111] *Dauzère-Pérès S., Lasserre J.B.* A note on Carlier's algorithm for the one-machine sequencing problem// Toulouse: CNRS, 1990.– Rapport LAAS N 90370.– 4 p.

[112] *Dauzère-Pérès S., Sevaux M.* Using Lagrangean relaxation to minimize the weighted number of late jobs on a single machine// Naval Res. Logistics.– 2003.– V. 50, N 3.– P. 273 – 288.

[113] *Dessouky M.I., Margenthaler C.R.* The one-machine sequencing problem with early starts and due dates// AIIE Trans.– 1972.– V. 4.– P. 214 – 222.

[114] *Du J., Leung J. Y.-T.* Minimizing total tardiness on one processor is NP-hard // Math. Operation Research. – 1990. – V.15. – P. 483–495.

[115] *Du J., Leung J.Y.-T., Young G.H.* Scheduling chain-structered tasks to minimize makespan and mean flow time. // Inform. and Comput. – 1991. – 92(2). – P. 219-236.

[116] *Elmaghraby S.E.* The one machine scheduling ptoblem with delay costs // Journal of Industrial Engineering. – 1968. – V.19. – P. 105–108.

[117] *Emmons H.* One machine sequencing to minimizing certain function of job tardiness // Operations Research. – 1969. – V.17. – P. 701–715.

[118] *Fisher M.L.* A dual problem for the one machine scheduling problem // Mathematics Programming. – 1976. – V.11. – P. 229–251.

[119] *Gafarov E.R., Lazarev A.A.* Graphical approach for solving combinatorial problems.// International Conference on Operations Research, – Karlsruhe, Germany, 6-8 September 2006, P. 59.

[120] *Gafarov E.R., Lazarev A.A.* Algorithms for single machine total tardiness problem $1||\sum T_j$.// International Conference on Operations Research. – Karlsruhe, Germany, 6-8 September 2006, P. 83

[121] *Gantt H.L.* ASME Transactions, 1903, 24, P. 1322–1336.

[122] *Garey M.R., Johnson D.S.* Scheduling tasks with nonuniform deadlines on two processors // J. Assoc. Comput. Mach. – 1976. – 23. – P. 461-467.

[123] *Garey M.R. and Johnson D.S.* Two-processor scheduling with start-times and deadlines // SIAM J. Comput. – 1977. – 6. — 3. – P. 416-426.

[124] *Garey, M.R., Johnson, D.S.* "Strong" NP-completeness results: motivation, examples, and implications // J. Assoc. Comput. Mach. – 1978. – **25, 3** – P. 499–508.

[125] *Graham R.L., Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G.* Optimization and approximation in deterministic sequencing and scheduling: a survey// Ann. Discrete Math.– 1979.– V. 5.– P. 287 – 326.

[126] *Guéret C., Prins C., Sevaux M., Heipcke S.* Applications of Optimization with XpressMP// Dash Optimization Ltd., 2002.– 264 p.

[127] *Hall L.A., Shmoys D.B.* Jackson's rule for one-machine schedulings: Making a good heuristic better// Math. Oper. Res.– 1992.– V. 17. –
P. 22 – 35.

[128] *Held M., Karp R.M.* A dynamic programming approach to sequencing problems // SIAM Journal. – 1962. – V.10. – P. 196–210.

[129] *Holsenback J.E., Russell R.M.* A heuristic algorithm for sequencing on one machine to minimize total tardiness // Journal of Operation Research Society. – 1992. – V.43. – P. 53–62.

[130] *Hoogeveen J. A.* Minimizing maximum promptness and maximum lateness on a single machine// Math. Oper. Res. – 1996. – V. 21. –
P. 100 – 114.

[131] *Hooker J.N., Ottosson G.* Logic-based Benders decomposition// Math. Prog.– 2003.– V. 96.– P. 33 – 60.

[132] *Hu, T.C.* Parallel sequencing and assembly line problems // Oper. Res.– 1961. – 9. – P. 841–848.

[133] *Jackson J.R.* Scheduling a production line to minimize maximum tardiness// Los Angeles, CA: University of California, 1955.– Manag. Sci. Res. Project. Research Report N 43.

[134] *Jain V., Grossmann I.E.* Algorithms for hybrid MILP/CLP models for a class of optimization problems// INFORMS J. Computing.– 2001.– V. 13.– P. 258 – 276.

[135] *Johnson S.M.* Optimal two- and three-stage production schedules with setup times included. // Naval Research Logistics Quarterly. – 1954. – V.1. – P. 61–68.

[136] *Kao E.P.C., Queyranne M.* On dynamic programming methods for assembly line balancing // Operations Research. – 1982. – V.30. – P. 375–390.

[137] *Kellerer H., Pferschy U., Pisinger D.* Knapsack problems, Springer, 2010, – 546 p.

[138] *Koulamas C.P.* The total tardiness problem: Review and extensions // Operations Research. – 1994. – V.42. – P. 1025–1041.

[139] *Kise H., Ibaraki T., Mine H.* A solvable case of the one machine scheduling problem with ready and due times// Oper. Res.– 1978.– V. 26, N 1.– P. 121 – 126.

[140] *Kubiak W.* Exact and approximate algorithms for scheduling unit time tasks with trre-like precedence constraints. // In Abstracts EURO IX - TIMS XXVIII Paris.– 1988.– P. 195.

[141] *Landsberg J. M.* Geometry and Complexity Theory|// Cambridge University Press.- 2016.

[142] *Lageweg B.J., Lenstra J.K., Rinnooy Kan A.H.G.* Minimizing maximum lateness on one machine: computational experience and applications// Statistica Neerlandica.– 1976.– V. 30.– P. 25 – 41.

[143] *Land A.H., Doig A.G.* An automatic method for solving discrete programming problems// Econometrica.– 1960.– V. 28.– P. 497 – 520.

[144] *Lawler E.L.* Optimal sequencing of a single machine subject to precedence constraints// Manag. Sci.– 1973.– V. 19, N 5.– P. 544 – 546.

[145] *Lawler E.L.* A dynamic programming algorithm for preemptive scheduling of a single machine to minimize the number of late jobs// Annals of Oper. Res.– 1990.– V. 26.– P. 125 – 133.

[146] *Lawler E.L.* Knapsack-like scheduling problems, the Moore-Hodgson algorithm and the "tower od sets" property// Math. Comp. Modelling.– 1994.– V. 20, N 2.– P. 91 – 106.

[147] *Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G., Shmoys D.B.* Sequencing and schedulinq: algorithms and complexity// Graves S.C., Rinnooy Kan A.H.G., Zipkin P.H. (eds.) Handbooks in Operations Research and Management Science.– Vol. 4, Logistics of Production and Inventory.– NewYork: NorthHolland, 1993.– P. 452 – 522.

[148] *Lawler E.L., Moore J.M.* A functional equation and its application to resource allocation and sequencing problems// Manag. Sci.– 1969.– V. 16.– P. 77 – 84.

[149] *Lawler E.L.* On scheduling problems with deferral costs // Management Science. – 1964. – V.11.- P. 280–288.

[150] *Lawler E.L.* A pseudopolynomial algorithm for sequencing jobs to minimize total tardiness // Ann. Discrete Math. – 1977. – V.1. – P. 331–342.

[151] *Lawler E.L.* A fully polynomial approximation scheme for the total tardiness problem // Oper. Res. Lett. – 1982. – V.1. – P. 207–208.

[152] *Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G., Shmoys D.B.* Sequencing and Schedling: Algorithms and Complexity // Elsevier Science Publishers B.V. – 1993. – V.4. – P. 445–520.

[153] *Lazarev A. A.* Analysis of Scheduling Problems Using Transformation.// Journal of Soviet Mathematics, Vol. 45, N 2, April 1989. P. 1029–1036.

[154] *Lazarev A.A.* Scheduling algorithms based on necessary optimality. // Journal of Soviet Mathematics, V 44, N 5, 1989, P. 635–642.

[155] *Lazarev A.* Scheduling to minimize maximum lateness for single machine: new approach of investigation// 9-th Belgian-French-German Conference on Optimization, Namur, September, 7-11, 1998,
http://www.fundp.ac.be/ bfgconf9/.

[156] *Lazarev A.A.* Analize of structure of optimal schedule the problem minimizing maximum lateness for single machine// the Fourth International Congress on Industrial and Applied Mathematics, 5-9 July 1999, Edinburgh, Scotland, p 283.

[157] *Lazarev A.A.* Minimum absolute error for NP-hard scheduling problem for single machine - minimizing maximum lateness// Workshop on the Complexity of Multivariate Problems, October 4-8, 1999, Hong Kong, China, p. 13.

[158] *Lazarev A.A., Gafarov E.R.* Special case of the single machine total tardiness problem is $NP-$hard.// 12th IFAC Symposium on Information Control Problems in Manufacturing. INCOM'2006. Preprints, Vol. III, Operational Research, May 17–19, 2006 – Saint-Etienne, France. – P. 155–157.

[159] *Lazarev A., Kvaratskhelia A.* Methods to research problem of scheduling theory minimizing total tardiness on a single machine. // In Book of

Abstracts of VI Workshop MAPSP'03 «Models an Algorithms for Planning and Scheduling Problems» (Aussois, 30.III - 4.IV.2003). – P. 143–144.

[160] *Lazarev A., Kvaratskhelia A.* Algorithms for solving problems $1 \mid\mid \sum Tj$ and Even-Odd Partition. // In Book of Abstracts of XVIII Internationa Conference «European Chapters on Combinatorial Optimization». – Minsk, 26–28.V.2005. – P. 32–33.

[161] *Lazarev A., Kvaratskhelia A., Tchernykh A.* Solution algorithms for the total tardiness scheduling problem on a single machine. // Workshop Proceedings of the ENC'04 Mexican International Conference in Computer Science. – Mexico, 2004. – P. 474–480.

[162] *Lazarev A.A., Sadykov R.R.* A polynomial approximation scheme for the $1 \mid r_j \mid L_{\max}$ scheduling problem with guaranteed absolute error// Arias Estrada M., Gelbukh A. (eds.) Avances en la Ciencia de la Computación: Proceedings of Fifth Mexican International Conference ENC'04, 20 – 24 september.– Colima, Mexico: 2004.– P. 465 – 473.

[163] *Lazarev A., Schul'gina O..* Problem minimizing maximum lateness for single machine: properties, procedures, algorithms// 9-th Belgian-French-German Conference on Optimization, Namur, September 7 – 11.– Namur, 1998.– P. 45.

[164] *Lazarev A., Siraev R.* Scheduling to minimize total weighted completion time: branch and bound method// 9-th Belgian-French-German Conference on Optimization, Namur, September, 7-11, 1998,
http://www.fundp.ac.be/ bfgconf9/.

[165] *Lazarev A.A., Sologub A.A., Korenev P.S.* A metric for total tardiness minimization // Automation and remote control. 2017. Vol.78,No. 4. C.732-740.

[166] *Lazarev A.A., Werner F.* Algorithms for Special Single Machine Total Tardiness Problem and an Application to the Even-Odd Partition Problem // Math. and Comp. Model. 2009. No. 49. 2078–2089.

[167] *Lenstra J.K., Rinnooy Kan A.H.G., Brucker P.* Complexity of machine scheduling problems// Annals of Discrete Math.– 1977.– V. 1.– P. 343 – 362.

[168] *Lenstra J.K., Rinnooy Kan A.H.G., Brucker P.* Complexity of machine scheduling problems // European Journal of Operational Research. – 1980. – V.4. – P. 270–275.

[169] *Leung, J.Y.-T. and Vornberger, O. and Witthoff, J.D.* On some variants of the bandwidth minimization problem // SIAM J. Comput., SIAM Journal on Computing. – 13 – 1984.– 3. – P. 650–667.

[170] *MacCormick J.* What Can Be Computed?: A Practical Guide to the Theory of Computation // Princeton University Press – 2018.

[171] *Marriott K., Stuckey P.J.* Programming with Constraints: An Introduction// The MIT Press, 1998.– 476 p.

[172] *Mastrolilli M.* Efficient Approximation Schemes for Scheduling Problems with Release Dates and Delivery Times// J. of Scheduling.– 2003.– V. 6, N 6.– P. 521 – 531.

[173] *McMahon G., Florian M.* On scheduling with ready times and due dates to minimize maximum lateness// Oper. Res.– 1975.– V. 23.– P. 475 – 482.

[174] *McNaughton R.* Scheduling with deadlines and loss functions // Management Science. – 1959. – V.6. – P. 1–12.

[175] *Morton T.E., Rachamadugu R.M., Vepsalainen A.* Accurate myopic heuristics for tardiness scheduling // GSIA Working Paper 36-83-84, Carnegie Mellon University. – 1984.

[176] *Moore J.M.* An n job, one machine sequencing algorithm for minimizing the number of late jobs// Manag. Sci.– 1968.– V. 15, N 1.– P. 102 – 109.

[177] em Nourani C.F. Algebraic Computability and Enumeration Models // Recursion Theory and Descriptive Complexity. Apple Academic Press – 2016.

[178] *Nowicki E., Zdrzalka S.* A note on minimizing maximum lateness in a one-machine sequencing problem with release dates// European J. of Oper. Res.– 1986.– V. 23.– P. 266 – 267.

[179] *Panwalkar S.S., Smith M.L., Koulamas C.P.* A heuristic for the single machine tardiness problem // European Journal of Operation Research. – 1993. – V.70. – p. 304–310.

[180] *Peridy, L., Pinson E., Rivreau D.* Using short-term memory to minimize the weighted number of late jobs on a single machine// European J. Oper. Res.– 2003.– V. 148. P. 591 – 603.

[181] *Picard J., Queyranne M.* The time-dependent travelling salesman problem and its application to the tardiness problem in one machine // Operations Research. – 1978. – V.26. – P. 86–110.

[182] *Pinedo M.* Scheduling: Theory, Algorithms, and Systems. – Prentice-Hall Englewood Cliffs, New Jersey. – 1995.

[183] *Potts C.N., Van Wassenhove L.N.* A decomposition algorithm for the single machine total tardiness problem // Oper. Res. Lett. – 1982. – V.5. – P. 177–182.

[184] *Potts C.N., Van Wassenhove L.N.* A branch-and-bound algorithm for the weighted tardiness problem // Operations Research. – 1985. – V.33. – P. 363–377.

[185] *Potts C.N., Van Wassenhove L.N.* Dynamic programming and decomposition approaches for the single machine total tardiness problem European Journal of Operational Research. – 1987. – V.32. – P. 405–414.

[186] *Potts C.N.* Analysis of a heuristic for one machine sequencing with release dates and delivery times// Oper. Res.– 1980.– V. 28. P. 1436 – 1441.

[187] *Potts C.N., Van Wassenhove L.N.* Single machine tardiness sequencing heuristics // IIE Transactions. – 1991. – V.23. – P. 93–108.

[188] *Rinnooy Kan A.H.G., Lageweg B.J., Lenstra J.K.* Minimizing total cost in one machine scheduling // Operations Research. – 1975. – V.23. – P. 908–927.

[189] *Queyranne M., Schultz A.S.* Polyhedral approaches to machine scheduling// Preprint N 408/1994.– Berlin: Technical University of Berlin, Department of Mathematics, 1994.– 61 p.

[190] *Sadykov R.R., Lazarev A.A.* Experimental comparison of branch-and-bound algorithms for the $1 \mid r_j \mid L_{\max}$ problem// Proceedings of the Seventh International Workshop MAPSP'05, 6 – 10 june.– Siena, Italy: 2005.– P. 239 – 241.

[191] *Sadykov R.R., Lazarev A.A., Kvaratskhelia A.G.* Research of absolute error for NP-hard scheduling problem minimizing maximum lateness// Proceedings of the Tenth Franch-German-Italian Conference on Optimization FGI'00, 4 – 8 september.– Montpelier, France: 2000. – P. 56 – 57.

[192] *Schrage, L.* Solving Resource-Constrained Network Problems by Implicit Enumeration: Non Preemptive Case// Oper. Res. – 1970. – V. 18. – P. 263 – 278.

[193] *Schild L., Fredman K.R.* Scheduling tasks with linear loss function // Management Science. – 1961. – V.7. – P. 280–285.

[194] *Schrage L., Baker K.R.* Dynamic programming solution of sequencing problems with precedence constraints // Operations Research. – 1978. – V.26. – P. 444–449.

[195] *Sen T., Austin L.M., Ghandforoush P.* An algorithm for the single machine sequencing problem to minimize total tardiness // IIE Transactions. – 1983. – V.15. – P. 363–366.

[196] *Sen T., Borah B.N.* On the single machine sheduling problem with tardiness penalties // Journal of Operational Research Society. – 1991. – V.42. – P. 695–702.

[197] *Sen T., Sulek J.M., Dileepan P.* Static scheduling research to minimize weighted and unweighted tardiness: A state-of-the-art survey. // International Jornal of Production Economics. – 2003. – V.83. – P. 1–12.

[198] *Sevastianov S.V., Tchernykh I.D.* Computer-Aided Way to Prove Theorems in Scheduling // Bilardi G., Italiano G.F., Pietracaptina A., Pucci G. (eds.) Proceedings of Sixth Annual European Symposium on Algorithms ESA'98, 24 – 26 august.– Venice, Italy: 1998.– Springer-Verlag, LNCS, V. 1461, 1998.– P. 502 – 513.

[199] *Sevastianov S.V., Woeginger G.J.* Makespan minimization in open shops: A polynomial time approximation scheme // Math. Prog.– 1998.– V. 82.– P. 191 – 198.

[200] *Sevaux M., Dauzère-Pérès S.* Genetic algorithms to minimize the weighted number of late jobs on a single machine// European J. of Oper. Res.– 2003.– V. 151.– P. 296 – 306.

[201] *Shwimer J.* On the n-job one machine sequence-independent scheduling problem with tardiness penalties // Management Science. – 1972. – V.18. – p. 301–313.

[202] *Simons B.B.* A fast algorithm for single processor scheduling// Proceedings of the 19th IEEE Annual Symposium on Foundations of Computer Science.– New York: Ann. Arbor. Mich., 1978.– P. 246 – 252.

[203] *Simons, B.* Multiprocessor scheduling of unit-time jobs with arbitrary release times and deadlines // SIAM J. Comput. 12, 1983, 2, P. 294–299.

[204] *Smith W.E.* Various optimizers for single stage production // Naval Research Logistics Quarterly. – 1956. – V.3. – P. 59–66.

[205] *Sousa J.P., Wolsey L.A.* A time-indexed formulation of non-preemptive single-machine scheduling problems// Math. Prog.– 1992.– V. 54.– P. 353 – 367.

[206] *Srinivasan V.* A hybrid algorithm for the one machine sequencing problem to minimize total tardiness // Naval Research Logistics Quaterly. – 1971. – V.18. – P. 317–327.

[207] *Szwarc W.* Single machine total tardiness problem revised // Creative and Innovate Approaches to the Science of Management, Quorum Books. – 1993. – P. 407–419.

[208] *Szwarc W., della Croce F., Grosso A.* Solution of the single machine total tardiness problem // Journal of Scheduling. – 1999. – V.2. – P. 55–71.

[209] *Szwarc W., Grosso A., Della Croce F.* Algorithmic paradoxes of the single machine total tardiness problem // Journal of Scheduling. – 2001. – V.4. – P. 93–104.

[210] *Szwarc W., Mikhopadhyay S.* Decomposition of the single machine total tardiness problem //Oper. Res. Lett. – 1996. – V.19. – P. 243–250.

[211] *Tansel B.C., Sabuncuoglu I.* New insights on the single machine total tardiness problem // Operations Research Letters. – 1997. – V.48. – P. 82–89.

[212] *Tansel B.C., Kara B.Y., Sabuncuoglu I.* An efficient algorithm for the single machine total tardiness problem // IIE Transactions. – 2001. – V.33. – P. 661–674.

[213] *Thorsteinsson E.S.* Branch-and-check: a hybrid framework integrating mixed integer programming and constraint logic programming// Proceedings of the Seventh International Conference CP'01, 26 november – 1 december.– Springer-Verlag, LNCS, V. 2239, 2001.– P. 16 – 30.

[214] *Timkovsky, V.G.* Identical parallel machines vs. unit-time shops and preemptions vs. chains in scheduling complexity, *European J. Oper. Res.*, 2003, 149, 2, 355-376.

[215] *Ullman J.D.* (1975) NP-complete scheduling problems // J. Comput. System Sci. – 1975. – **10**. P. 384–393.

[216] *Wilkerson L.J., Irvine J.D.* An improved algorithm for scheduling independent tasks // AIIE Transactions. – 1971. – V.3. – P. 239–245.