Лекция 6. Задачи планирования работ по проекту с учетом ограниченности ресурса

Сетевые графики. Задача определения продолжительности проекта. Расчет критического пути. Метод PERT (Program Evaluation and Review Technique).

Алгоритм диспетчеризации для задачи распределения ресурса. Нижние оценки для задачи распределения ресурсов и ее частных случаев. Точный алгоритм решения задачи распределения ресурсов методом ветвей и границ.

Эвристические алгоритмы решения задачи распределения ресурсов: распределение по критичности, по позднему времени окончания, на работы с минимальной длительностью. Оценки сравнительной эффективности эвристик. Алгоритм муравьиных колоний

Классификация задач распределения ресурсов между задачами проекта. Задача Джонсона. Задача упаковки в полосу и задача LSPP. Задача упаковки в палеты и задача UPT. Другие частные случаи.

Предисловие

- Три уровня задач управления проектами
  - Задачи мультипроектного управления (проект как единое целое)
  - Задачи календарно-сетевого планирования, то есть планирования последовательности выполнения работ в рамках одного проекта (работа рассматривается как единое целое)
  - Задачи управления интенсивностью работ (можно менять параметры отдельной работы)

Теория календарно-сетевого планирования: основные определения

- **Проект** — это уникальный набор процессов, состоящих из скоординированных и управляемых задач с начальной и конечной датами, предпринятых для достижения цели. Достижение цели проекта требует получения результатов, соответствующих определённым заранее требованиям, в том числе ограничения на получения результатов, таких как время, деньги и ресурсы. (ISO 21500)
- **Управление проектом** – это **планирование**, координация и контроль работ по проекту для достижения его целей в рамках заданного бюджета и сроков, с надлежащим качеством. («Консалтинг ПРИМ»)
- Одним из видов планирования в рамках проектного управления является планирование сроков начала работ и распределения между ними ресурсов – календарно-сетевое планирование.

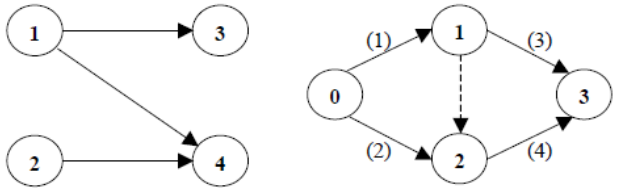
История теории календарно-сетевого планирования

- Истоки проектного планирования
  - М. Уолкер (компания Дюпон) и Д. Келли (Ремингтон Рэнд) – метод критического пути
  - Фирма Локхид и фирма «Буз, Аллен анд Гамильтон» ВМС США – метод PERT (Program Evaluation and Review Technique)
- Особенно широко методы календарно-сетевого планирования применяются в строительных проектах

См. Буркова И.В. *МЕТОД ДИХОТОМИЧЕСКОГО ПРОГРАММИРОВАНИЯ В ЗАДАЧАХ УПРАВЛЕНИЯ ПРОЕКТАМИ.* Диссертация на соискание ученой степени к.т.н. Москва, 2004

Сетевые графики

- При постановке задач ресурсного планирования предполагается, что проект описан в виде *комплекса работ* в определенными зависимостями между ними. Зависимости между работами отображаются в виде *сетевого графика* (сети).
- Сетевой график
  - Сеть – это ориентированный граф с выделенной вершиной – истоком и выделенной вершиной – стоком.



  - Вариант 1. Вершины сети соответствуют работам проекта
    - Дуги описывают обязательные ограничения предшествования, если есть дуга ij, то работа j не может начаться раньше окончания работы i.
  - Вариант 2. Вершины сети соответствуют событиям
    - Дуги соответствуют работам. Событие j не может наступить, если не выполнены все работы ij.
    - Для описания всех зависимостей между работами могут потребоваться дополнительные, фиктивные дуги.

Сетевые графики

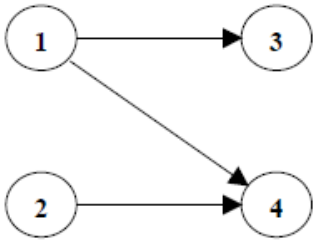
- Каждой работе соответствуют числовая характеристика - объем. Он измеряется в единицах трудоемкости или стоимости.
- Время выполнения работы зависит от интенсивности ее выполнения.
- В простейшем случае считается, что работа выполняется **с фиксированной интенсивностью**. Тогда работа характеризуется временем ее выполнения и объемом ресурсов различных видов (трудовых, оборудования, финансов и других) требуемых для ее выполнения.

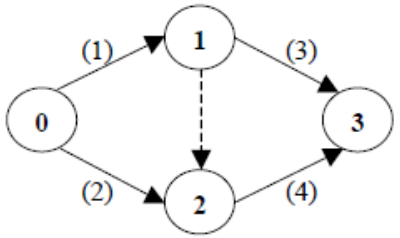
- Ресурсы делятся на возобновляемые (люди, оборудование) и невозобновляемые (материалы, финансы, …)

См. Буркова И.В. *МЕТОД ДИХОТОМИЧЕСКОГО ПРОГРАММИРОВАНИЯ В ЗАДАЧАХ УПРАВЛЕНИЯ ПРОЕКТАМИ.* Диссертация на соискание ученой степени к.т.н. Москва, 2004

Планирование работ по проекту. Расчет критического пути

- Начинаем с истока, присваивая ему метку «раннее время начала» $t_1 = 0$.
- Добавляем исток во «фронт работ» F
- Последовательно просматриваем все работы фронта работ
- Для каждой работы $i \in F$ последовательно перебираем ее последователей.





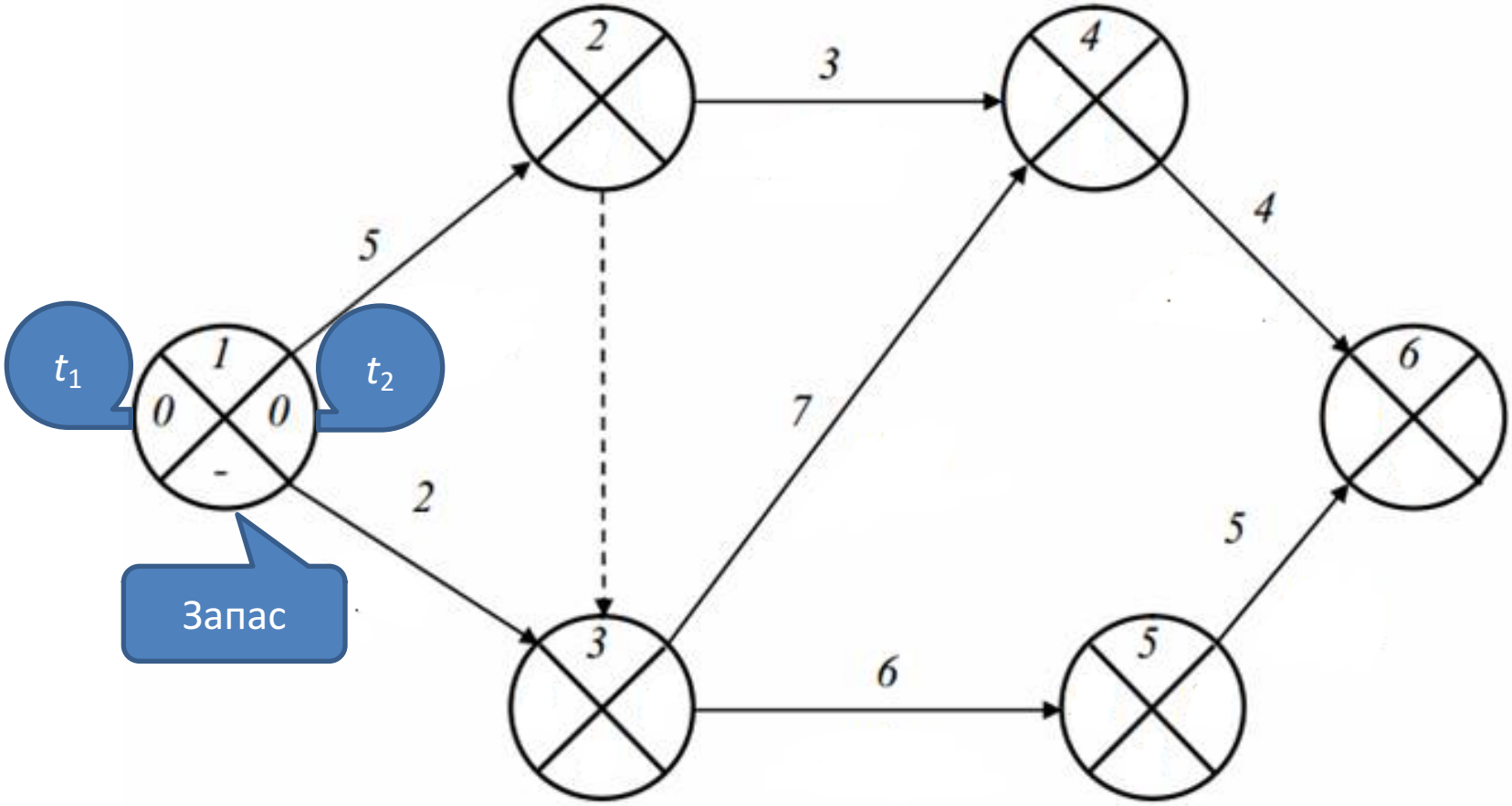
- Для каждого последователя $j \in F$, устанавливаем раннее время начала $t_1(j) = \max\{t_1(j), t_1(i) + p_i\}$.
- Добавляем последователя *j* во фронт работ *F*.
- Удаляем работу *i* из фронта работ.
- По мере исчерпания фронта работ останавливаемся.
- Раннее время начала для вершины-стока равно минимальному времени выполнения проекта (makespan).

Планирование работ по проекту. Расчет критического пути

- Критический путь можно восстановить обратным ходом
- Для этого повторяем вышеописанный алгоритм, но:
  - идя от стока назад по дугам
  - Заполняя «позднее время начала работы» $t_2(j)$
  - Присваивая стоку $t_2 = t_1$.
  - Для остальных работ - по формуле $t_2(j) = \min\{t_2(j), t_2(i) - p_j\}$.
- Разница t_2-t_1 для работы называется запасом. Критическому пути принадлежат работы с нулевым запасом и только они.
- Критический путь имеет максимальную продолжительность из всех путей, соединяющих исток со стоком

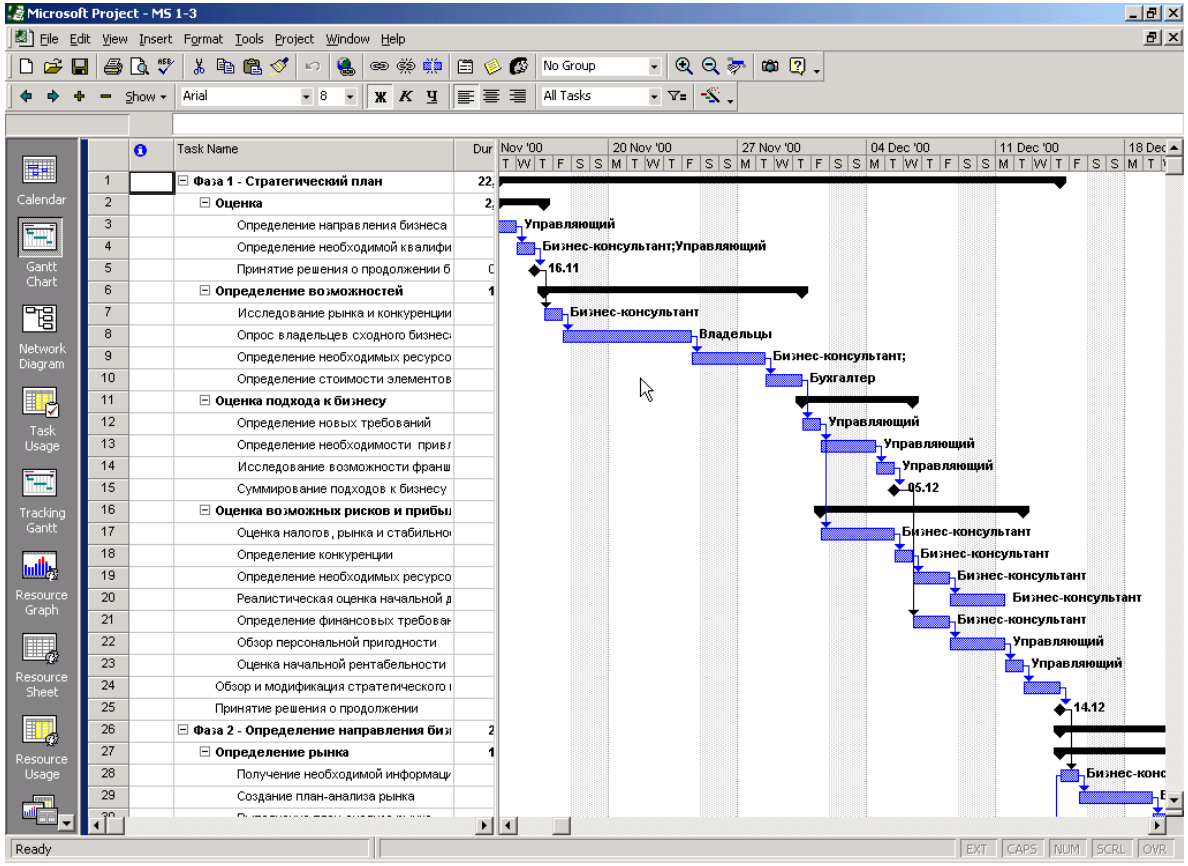
Планирование работ по проекту. Расчет критического пути. Пример

- Представление вершина-событие



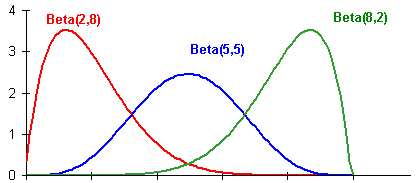
Планирование работ по проекту. Расчет критического пути

- Диаграмма Ганта – удобный способ отображения сетевого графика и построения критического пути (Г. Л. Гант, 1910 !).



Планирование работ по проекту. Метод PERT

- Для каждой работы
  - Пессимистическое время ее выполнения t_{max}
  - Оптимистическое время ее выполнения t_{min}
  - Наиболее вероятное время t_{mid}.
- Среднее время выполнения работы $t_{mean}=(t_{min} + 4 * t_{mid} + t_{max})/6$. примерно
- Среднеквадратичное отклонение времени выполнения работы $\sigma = (t_{max} - t_{min})/6$. еще более примерно.
- Какое распределение берется за основу?
- Как посчитать среднее время выполнения всего проекта?
- Как посчитать среднеквадратичное отклонение времени выполнения проекта?
- Как посчитать вероятность выполнения проекта к заданной дате?
- Как найти вероятность задачи оказаться на критическом пути?



- Oracle risk analyzer user manual

Задача построения расписания для проекта: базовая постановка

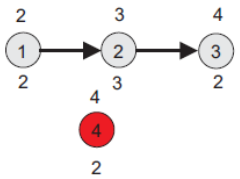
- Какого важного с точки зрения практики элемента не хватает в схеме расчета критического пути?
- Задача построения расписания для проекта (планирования работ по проекту, project scheduling, календарно-сетевого планирования)
- Базовая постановка: минимизация времени выполнения проекта (makespan) с учетом ограничения на ресурсы (resource-constrained project scheduling problem, RCPSP)

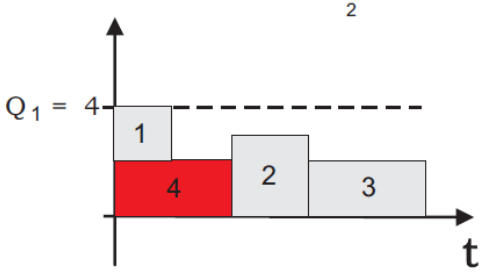
Минимизация времени выполнения проекта с учетом ограничения на ресурсы

- Задан сетевой график с фиксированными продолжительностями работ p_i и потребностями в возобновляемых ресурсах одного из K типов q_{ik}, i = 1, …, n, k = 1, …, K.
- В каждый момент времени *k*-й ресурс доступен в объеме Q_k
- Необходимо назначить моменты времени начала каждой работы S_i так, чтобы минимизировать время реализации проекта C^*_{max}:=max[S_i + p_i] при соблюдении следующих условий:
  - В каждый момент времени

$$\sum_{i:=1}^{n} q_{ik}\varphi_i(t) \le Q_k,\ k = 1,\dots,K$$

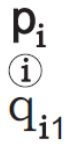
    где $\varphi_i(t)$ – индикатор выполнения задачи *i* в момент времени *t*
  - Если *i* -> *j*, то $S_i+p_i <= S_j$
- Время обычно дискретное (p_i целые)

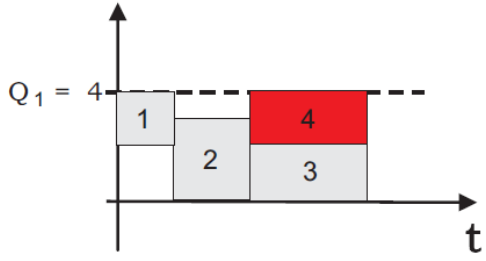




Допустимое расписание

C_{max}= 4+3+4 = 11





Оптимальное расписание

C_{max}= 2+3+4 = 9

Литература: Лазарев А.А. Гафаров Е.Р. Теория расписаний. М.: МГУ, 2011.

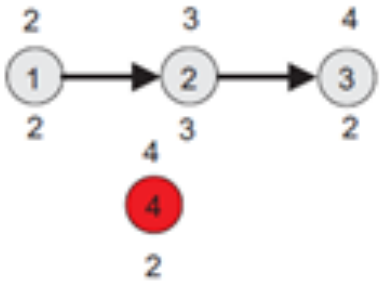
Задача построения расписания для проекта: краткая характеристика

- Пусть *T* – длина критического пути.
- Тогда $C^*_{max} >= T$.

- Пусть UB – верхняя оценка времени выполнения проекта (предложите тривиальную верхнюю оценку!)
- Тогда в любом оптимальном графике $S_i \geq r_i := t_1(i)$ – работа начинается не ранее раннего времени начала, $S_i \leq d_i := UB - T + t_2(i)$.
- r_i и d_i можно уточнять с уточнением UB. Например, любое допустимое решение дает верхнюю оценку.
- RCPSP даже в базовой постановке является NP-трудной
- Даже для одного ресурса неизвестны алгоритмы с фиксированной гарантированной относительной погрешностью
- Как обычно для NP-трудных задач
  - Ищутся нижние оценки
  - Рассматриваются модификации схемы ветвей и границ
  - Предлагаются эвристики
  - Рассматриваются частные случаи, в которых задача имеет (более) эффективное решение

Задача построения расписания для проекта: алгоритм диспетчеризации (LS, list scheduling)

1: Пусть EL – список всех требований без предшественников. Полагаем $Q_k(\tau) = Q_k, \forall \tau, k = 1, \ldots, K$.
2: **while** $EL \neq \emptyset$ **do**
3: Выберем требование $j \in EL$;
4: $t := \max\limits_{i|(i,j)\in A} \{S_i + p_i\}$. Если для требования j предшественники не определены, тогда $t = 0$;
5: **while** Существует такой ресурс k, что $q_{jk} > Q_k(\tau)$ для некоторого $\tau \in [t, t+p_j)$ **do**
6: Вычислим такое минимальное значение $t_k > t$, что требование j может быть обслужено на интервале $[t_k, t_k + p_j)$, если рассматривается только ресурс k;
7: $t := t_k$;
8: **end while**
9: Назначим выполнение требования j на интервал $[S_j, C_j) = [t, t + p_j)$;
10: Резервируем ресурсы под выполнение требования j: $Q_k(\tau) = Q_k(\tau) - q_{jk},\ k = 1, \ldots, K,\ \tau \in [t, t + p_j)$;
11: $EL = EL \setminus \{j\}$;
12: Добавляем в EL последователей требования j, для которых все предшественники поставлены в расписание;
13: **end while**



- Трудоемкость алгоритма $O(n^2K)$ операций.
- По нашей классификации, к какому классу относится этот алгоритм?
- Полностью ли он определен, или есть некоторый простор для различных трактовок?

Задача построения расписания для проекта: алгоритм диспетчеризации (LS, list scheduling)

- *Активным* назовем такое допустимое расписание (решение) $S = (S_1, \ldots, S_n)$, для которого не существует другого допустимого расписания $S' = (S_1', \ldots, S_n')$ такого, что $S_j' \le S_j$, $\forall j \in N$, и хотя бы одно из неравенств строгое.
- Как понятие активного расписания связано с понятием оптимальности по Парето?

- Алгоритм LS строит только активные расписания.
- Отсюда вывод – локальные улучшения решения LS неочевидны.
- В то же время, оптимальное расписание активно.

Задача построения расписания для проекта: алгоритм диспетчеризации (LS, list scheduling)

- Теорема 5.1 (из Лазарев, Гафаров, 2011) *Активное расписание однозначно представляется в виде перестановки* $\pi = (j_1, \ldots, j_n)$ *n требований.*
- Перестановка π задает последовательность выбора требований j на шаге 3 алгоритма LS.
- Таким образом, варьируя правила выбора требования на шаге 3 алгоритма LS, можно получить любое активное расписание, в том числе, и оптимальное.
- Поэтому подбор правила выбора требования (задачи проекта) в LS является полезным и увлекательным занятием (которым мы займемся чуть позднее).

Задача построения расписания для проекта: алгоритм диспетчеризации (LS, list scheduling)

- На основании LS можно построить точный алгоритм Ветвей и Границ.
- Ветвление происходит при выборе требования *j*. Что еще нужно поменять в алгоритме?

1: Пусть EL – список всех требований без предшественников. Полагаем $Q_k(\tau) = Q_k, \forall \tau, k = 1, \ldots, K$.
2: **while** $EL \neq \emptyset$ **do**
3: Выберем требование $j \in EL$;
4: $t := \max\limits_{i|(i,j)\in A} \{S_i + p_i\}$. Если для требования j предшественники не определены, тогда $t = 0$;
5: **while** Существует такой ресурс k, что $q_{jk} > Q_k(\tau)$ для некоторого $\tau \in [t, t+p_j)$ **do**
6: Вычислим такое минимальное значение $t_k > t$, что требование j может быть обслужено на интервале $[t_k, t_k + p_j)$, если рассматривается только ресурс k;
7: $t := t_k$;
8: **end while**
9: Назначим выполнение требования j на интервал $[S_j, C_j) = [t, t + p_j)$;
10: Резервируем ресурсы под выполнение требования j: $Q_k(\tau) = Q_k(\tau) - q_{jk},\ k = 1, \ldots, K,\ \tau \in [t, t + p_j)$;
11: $EL = EL \setminus \{j\}$;
12: Добавляем в EL последователей требования j, для которых все предшественники поставлены в расписание;
13: **end while**

- Так как алгоритм Ветвей и Границ является одним из наиболее эффективных точных алгоритмов для решения данной задачи, то становится актуальным нахождение “хороших” нижних и верхних оценок.

Нижние оценки RCPSP

- Задача с разрешенными прерываниями работ, PRCPSP
  - По очевидным причинам ее решение дает нижнюю оценку исходной задачи
  - Можно привести пример (какой?), когда оценка в 2 раза меньше оптимального решения
  - Эту оценку еще саму посчитать как-то нужно
- Длина критического пути T, или LB0 - еще одна оценка снизу
  - Игнорирует дефицит ресурса
  - Может быть в n раз меньше оптимального решения (пример!)
  - Находится за n^2 операций

Нижние оценки RCPSP

- Оценка, основанная на полном использовании максимально дефицитного ресурса, $LB_1 = \max_k \sum_j q_{jk} p_j / Q_k$.
  - Игнорирует последовательность работ
  - Считается за *n K* операций
  - Разница между решением и LB1 может достигать почти UB
- Дополнение критического пути, LB_S
  - Для каждой работы *i* не из критического пути вычисляется время τ_i, в течение которого она может выполняться одновременно с работами критического пути, не нарушая ресурсных ограничений. Тогда $LB_s = T + \max_i [p_i - \tau_i]$.

Нижняя оценка Буркова (1972)-Mingozzi (1998)

Множество требований $X \subset N$ будем называть *допустимым множеством*, если между каждой парой требований $i, j \in X$ нет отношений предшествования (прямых или косвенных) и требования могут обслуживаться параллельно без нарушения ресурсных ограничений ($\sum\limits_{i \in X} q_{ik} \leq Q_k, k = 1, \ldots, K$).

Допустимое множество X назовем *доминирующим*, если не существует другого допустимого множества Y такого, что выполняется $X \subset Y$.

Рассмотрим список всех доминирующих множеств $X_1, \ldots, X_f$ и соответствующие им векторы $a^j \in \{0,1\}^n,\ j = 1, \ldots, f$: $a_i^j = 1$, если $i \in X_j$, иначе $a_i^j = 0,\ i = 1, \ldots, n$.

Определим через x_j длину интервала, в котором все требования множества X_j обслуживаются параллельно. Тогда решение следующей задачи линейного программирования позволяет вычислить нижнюю оценку LB_M для $C^*_{\max}$ [86]:

$$\begin{cases} \min \sum\limits_{j=1}^{f} x_j, \\ \sum\limits_{j=1}^{f} a_i^j x_j \geq p_i,\ i = 1, \ldots, n, \\ x_j \geq 0,\ j = 1, \ldots, f. \end{cases}$$

Нижняя оценка Буркова-Mingozzi

- Проблемы
  - Число доминирующих множеств быстро растет с ростом числа задач *n*
  - Качество оценки по-прежнему «плохое», хотя лучше, чем у других оценок
  - И, наконец,

Лемма 5.4 *Вычисление оценки LB_M является NP–трудной задачей.*

Доказательство. Рассмотрим NP-трудную задачу УПАКОВКИ В ПАЛЛЕТЫ. Паллеты – это "коробки", в которые по ширине и высоте помещается только один предмет. Даны стандартные паллеты длиной W и n предметов, каждый длиной w_i, $i = 1, \ldots, n$. Необходимо упаковать все предметы в паллеты так, чтобы число использованных паллет было минимальным.

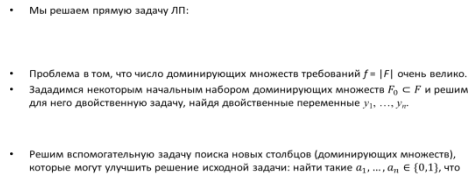
Преобразуем исходную задачу УПАКОВКИ В ПАЛЛЕТЫ в задачу RCPSP. Каждое требование $i = 1, \ldots, n$, соответствует предмету i. Зададим $p_i = 1$, $q_{i1} = w_i$, $i = 1, \ldots, n$, $Q_1 = W$. Отношения предшествования между требованиями не заданы. Тогда $C_{\max}(S^*)$ соответствует минимальному необходимому количеству паллет в исходной задаче.

Очевидно, что $LB_M = C^*_{\max}$ для построенного примера RCPSP. Следовательно, задача нахождения оценки LB_M эквивалентна решению NP-трудной задачи упаковки в паллеты. □

  - Пример для задачи с прерываниями показывает, что LB_M может быть в 2 раза меньше оптимального решения
  - можно подобрать также пример где LB_M в 2.5 раз меньше оптимального решения

Вычисление нижней оценки Буркова-Mingozzi методом (отложенной) генерации столбцов

- Мы решаем прямую задачу ЛП:



- Проблема в том, что число доминирующих множеств требований $f = |F|$ очень велико.
- Зададимся некоторым начальным набором доминирующих множеств $F_0 \subset F$ и решим для него двойственную задачу, найдя двойственные переменные $y_1, \ldots, y_n$.

Двойственная задача ЛП Для задачи $c^T x \to \min$ при $Ax \le b, x_i \ge 0, i = 1, \ldots, n$
двойственная задача $b^T y \to \max$ при $A^T y \ge c, y_i \ge 0, i = 1 \ldots m$

- Решим вспомогательную задачу поиска новых столбцов (доминирующих множеств), которые могут улучшить решение исходной задачи: найти такие $a_1, \ldots, a_n \in \{0,1\}$, что

$$\sum_{i=1}^{n} y_i \cdot a_i \to \max_{a\_1, \ldots, a\_n}$$

Критерий двойственной задачи

$$\sum_{i=1}^{n} r_{ik} \cdot a_i \le Q_k \quad (k = 1, \ldots, r)$$

Ресурсные ограничения

$$a_i \cdot a_j = 0 \quad (i - j \in D, i \to j \in C \text{ or } j \to i \in C).$$

Ограничения предшествования

Что это за задача?

Каким методом она решается?

Вычисление нижней оценки Буркова-Mingozzi методом (отложенной) генерации столбцов

- Найденные $a_1, \dots, a_n \in \{0,1\}$ дают новое доминирующее множество.
- Добавляем его в набор F_0.
- Если F_0 стал слишком велик, удаляем одно из старых доминирующих множеств (любое)
- Пересчитываем решение двойственной задачи

$$\sum_{i=1}^{n} y_i \cdot a_i > 1$$

- Как решить вспомогательную задачу?
- Здесь нужно «настоящее» дискретное решение – релаксации не подходят…
- Авторы [1] предлагают решать ее алгоритмом ветвей и границ.

$$\sum_{i=1}^{n} r_{ik} \cdot a_i \leq R_k \quad (k = 1, \dots, r)$$

$$a_i \cdot a_j = 0 \quad (i - j \in D, i \to j \in C \text{ or } j \to i \in C).$$

При ветвлении переменные a_i отбираются в порядке убывания y_i. При каждом ветвлении проверяются ресурсные ограничения и ограничения предшествования, так что все листья соответствуют допустимым множествам требований.

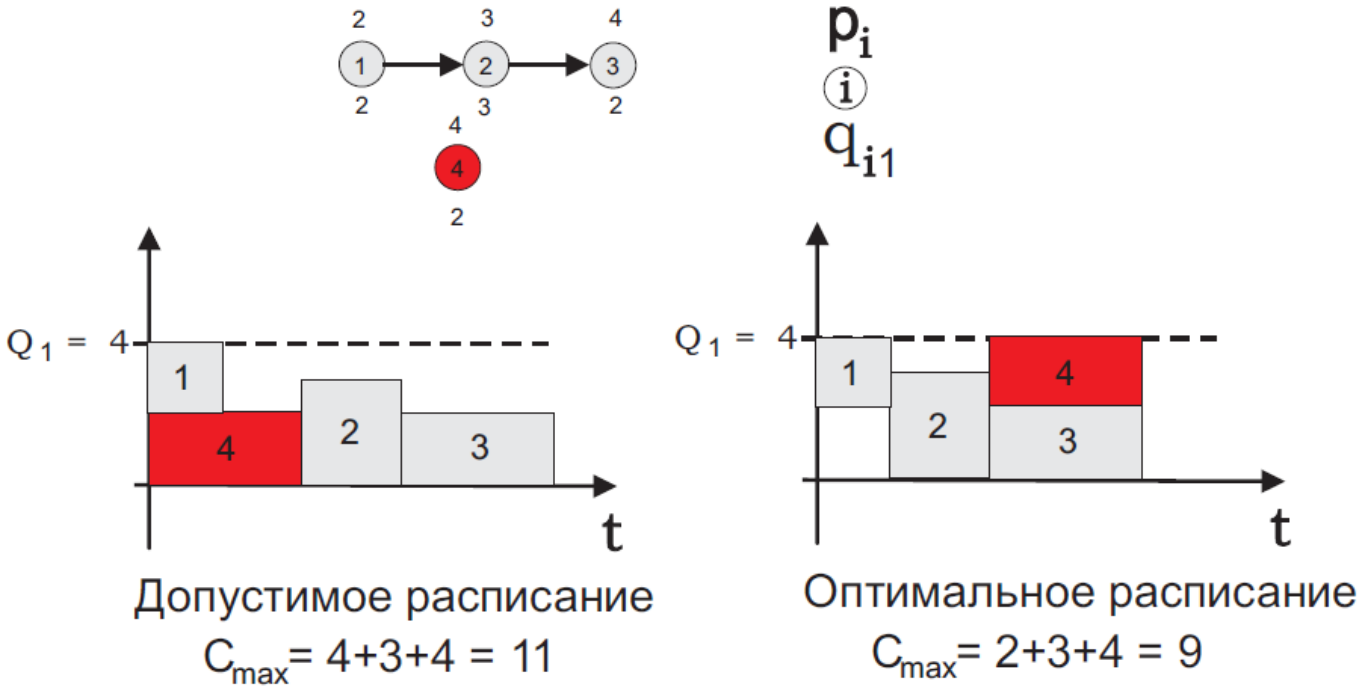
- Правило ограничения (bound) перебора:

$$\sum_{\lambda=1}^{k-1} y_\lambda \cdot a_\lambda + \sum_{\lambda=k+1}^{n_0} y_\lambda \leq \sum_{i=1}^{n_0} y_i \cdot a_i^*$$

- [1] *Tonius Baar, Peter Brucker and Sigrid Knust.* Tabu Search Algorithms and Lower Bounds for the Resource-Constrained Project Scheduling Problem. In “Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization”. Springer, 1999.

Правила диспетчеризации - верхние оценки для задачи ресурсного планирования

- Какие предложения?



Правила диспетчеризации - верхние оценки для задачи ресурсного планирования

Обозн-ие	Описание	Отн. погрешн.
Правила, основанные на параметрах требований		
SPT	выбрать работу с наименьшей продолжительностью обслуживания	$O(n)$
LPT	выбрать работу с наибольшей продолжительностью обслуживания	

Обозн-ие	Описание	Отн. погрешн.
Правила, основанные на отношениях предшествования		
MIS	выбрать работу с наибольшим количеством непосредственных последователей	$O(\sqrt{n})$
LIS	выбрать работу с наименьшим количеством непосредственных последователей	
MTS	выбрать работу с наибольшим количеством всех последователей	
LTS	выбрать работу с наименьшим количеством всех последователей	
GRPW	выбрать работу с наибольшим весом, т.е. требование с наибольшой суммарной продолжительностью всех последователей	$O(n)$

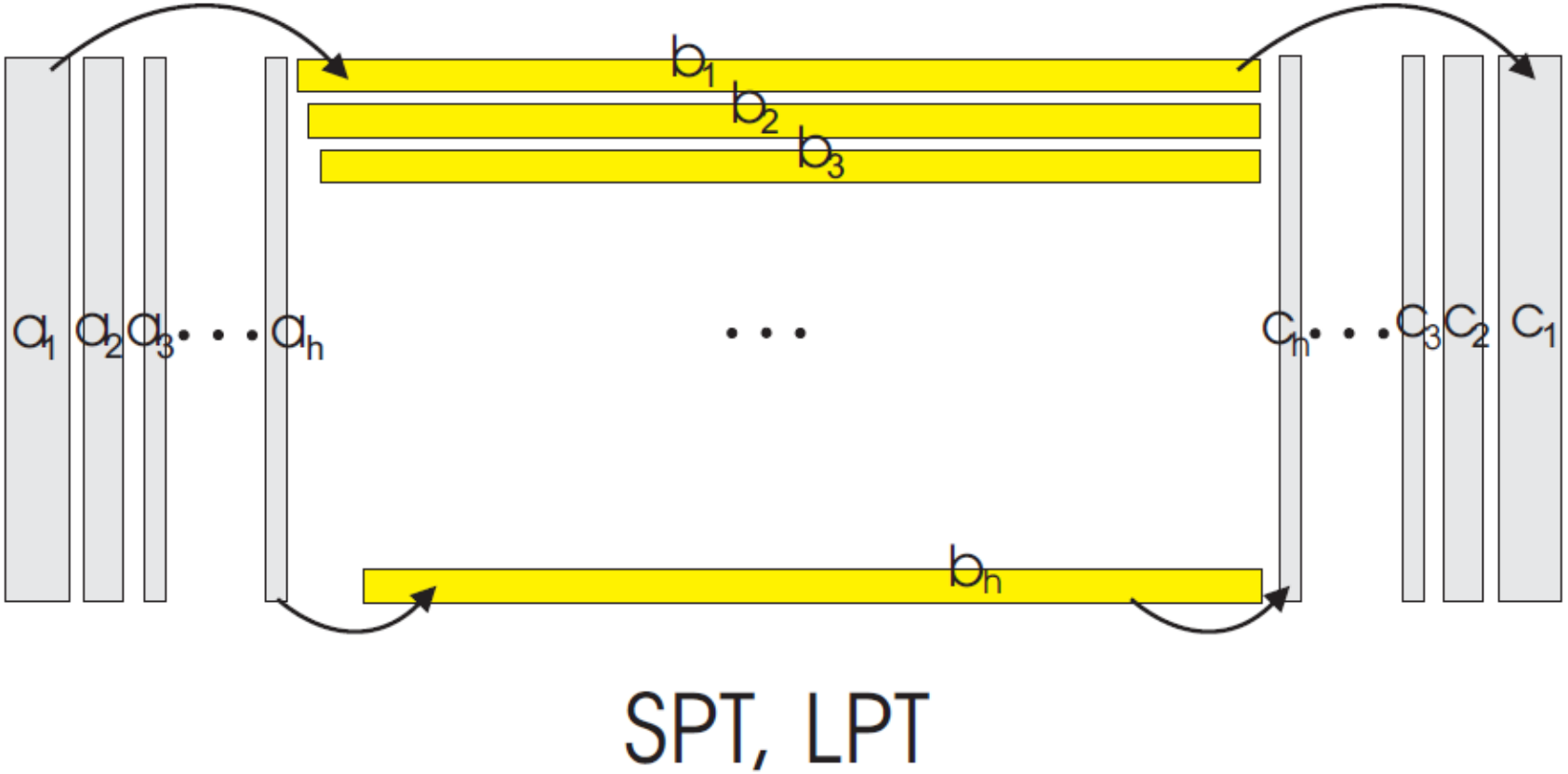
Правила диспетчеризации (продолжение)

Обозн-ие	Описание	Отн. по-грешн.
Правила, основанные на информации о критическом пути		
EST	выбрать работу с наим. возможным моментом начала r_i	$O(n)$
ECT	выбрать работу с наименьшим значением $r_i + p_i$	
LST	выбрать работу с наименьшим значением $d_i - p_i$	
LCT	выбрать работу с наименьшим значением d_i	
MSLK	выбрать работу с наименьшим ресурсом $d_i - r_i - p_i$,	
MW	выбрать работу с наименьшим окном $d_i - r_i$,	
DEST (Дина-мический EST)	выбрать работу с наим. возможным моментом начала r_i	
DECT (Дина-мический ECT)	выбрать работу с наименьшим значением $r_i + p_i$, где r_i вычисляется динамически	$O(\sqrt{n})$

Обозн-ие	Описание	Отн. по-грешн.
Правила, основанные на информации о необходимых ресурсах		
GRR	выбрать работу с наибольшей потребностью в ресурсах	$O(n)$

Правила диспетчеризации – примеры доказательств относительной погрешности

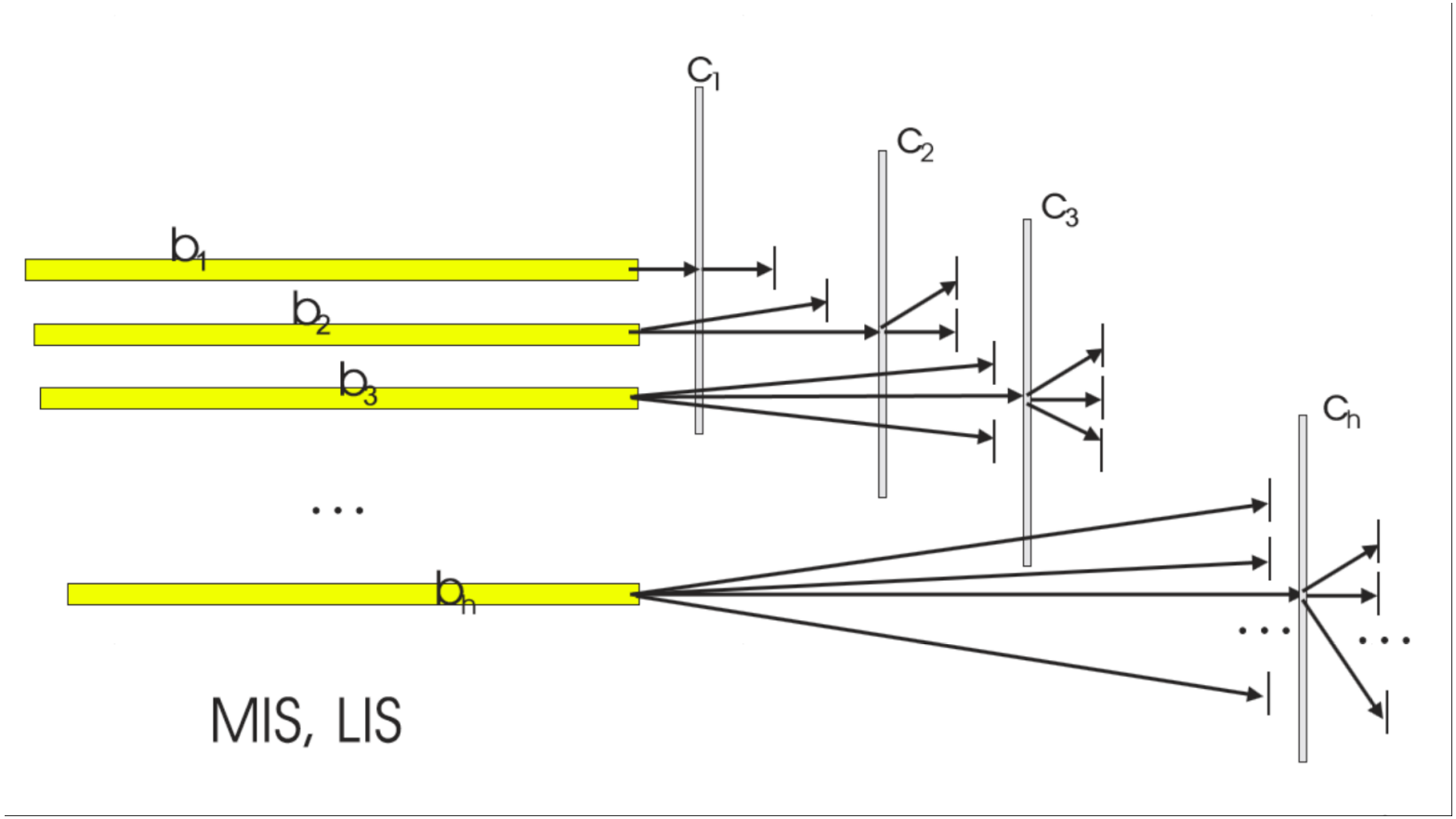
Теорема. Существует пример задачи распределения ресурсов проекта, для которой $C_{max}(LS_{SPT}) / C_{max}^{*} = O(n)$, где n – количество задач.



SPT, LPT

Правила диспетчеризации – примеры доказательств относительной погрешности

Теорема. Существует пример задачи распределения ресурсов проекта, для которой $C_{max}(LS_{MIS}) / C_{max}^{*} = O(\sqrt{n})$, где n – количество задач.



Алгоритм муравьиных колоний решения задачи распределения ресурсов по проекту

- Компоненты метода
  - Неориентированный или ориентированный взвешенный граф путей (решение задачи – маршрут в этом графе)
  - Правило выбора стартовой точки
  - Правило выбора пути из r в u $$P = \frac{\tau(r,u)^{\alpha} \times \eta(r,u)^{\beta}}{\sum_k \tau(r,u)^{\alpha} \times \eta(r,u)^{\beta}}.$$
  - Правило оценки маршрутов $$\Delta\tau_{ij}^{k}(t) = \frac{Q}{L^{k}(t)}.$$
  - Правило распределения феромона $$\tau_{ij}(t) = \Delta\tau_{ij}(t) + (\tau_{ij}^{k}(t) \times \rho).$$
  - Правило испарения феромона $$\tau_{ij}(t) = \tau_{ij}(t) \times (1 - \rho).$$

Частные случаи задачи распределения ресурсов по проекту

- Как мы видим, задача сложна, и нет оценок хорошего качества (даже нет верхних оценок с гарантированной относительной погрешностью)
- Но они есть для нескольких частных случаев с одним ресурсом
- Рассмотрим из для того, чтобы понять, как задача RCPSP связана с другими задачами теории расписаний
  - LSPP (like strip packing problem)
    - Нет отношений предшествования между работами
    - Похожа на задачу упаковки в полосу (потому и названа «like strip packing problem»)
  - UPT (unit processing time)
    - Единичные времена обслуживания
  - PMS (parallel machine scheduling)
    - Все потребности в ресурсе равны единице, количество ресурса $Q > 1$.

LSPP

Теорема 5.5 *Для частного случая LSPP выполняется*

$$\frac{C_{\max}(LS_{DEST})}{C^*_{\max}} < 3.$$

Доказательство. Пусть $\rho(t)$ (см. рис. 5.8) – уровень загрузки ресурса в момент времени t, т.е. $\rho(t) = \sum\limits_{i \in N_t} q_i$, где N_t – множество требований, обслуживаемых в момент времени t.

Обозначим $UB = C_{\max}(LS_{DEST})$. Пусть $[t_1, t_2) \in [0, UB)$ – интервал минимальной длинны, для которого выполняется: $\rho(t) > \frac{Q_1}{2}$ для всех $t \in [0, t_1)$, и $\rho(t) > \frac{Q_1}{2}$ для всех $t \in (t_2, UB)$.

Очевидно, что для расписания, построенного алгоритмом LS с правилом предпочтения DEST, мы имеем $t_2 - t_1 \le p_{\max}$. Иначе, если $t_2 - t_1 > p_{\max}$, тогда существует требование $j \in N_{t_2}$ с $q_j \le \frac{Q_1}{2}$ и $S_j > t_1$, т.е. получено противоречие, т.к. в соответствие с правилом предпочтения DEST, мы должны были поставить требование j на обслуживание с момента времени t_1. Значит, выполняется $t_2 - t_1 \le p_{\max}$. Поэтому $UB - p_{\max} < 2 \cdot \frac{\sum\limits_{i=1}^{n} q_i p_i}{Q_1}$. Более того, выполняется

$$C^*_{\max} \ge \frac{\sum\limits_{i=1}^{n} q_i p_i}{Q_1}$$

и

$$C^*_{\max} \ge p_{\max}.$$

Значит $UB < 3C^*_{\max}$. □

Кстати, докажите, что $C^*_{\max} \le 3 \max\{LB_0, LB_1\}$

PMS

Теорема 5.7 *Для частного случая PMS выполняется*

$$\frac{C_{\max}(LS_{DEST})}{C^*_{\max}} \leq 2.$$

Доказательство. Доказательство данной теоремы аналогично доказательству теоремы 5.5. Достаточно показать, что суммарная длина промежутков времени, где не все приборы заняты, меньше или равна длине критического пути (т.е. нижней оценке LB_0). Тогда можно будет доказать, что $C_{\max}(LS_{DEST}) \leq LB_0 + \frac{\sum p_j}{m} \leq 2C^*_{\max}$.

UPT

- Можно свести к PMS и показать, что $\frac{C_{\max}(LS_{EST})}{C^*_{\max}} < 4.$

Задача Джонсона и распределение SPT на работу с минимальной продолжительностью

- Двухприборная задача – поток независимых требований, каждое из которых должно сначала обрабатываться первым прибором (время обработки a_i) – а потом вторым (время обработки b_i)
- Порядок обслуживания на приборах одинаков
- Алгоритм Джонсона сложности n ln n. См. Лазарев-Гафаров, с. 144.
- Оптимально распределять при $a_i < b_i$ в порядке возрастания a_i, а для $a_i > b_i$ в порядке убывания b_i
- Какое отношение к распределению SPT и LPT?