Лекция 3. Алгоритмические методы решения задач дискретной оптимизации

Общая постановка задачи дискретной оптимизации. Классификация методов решения. Точные методы. Приближенные методы.

Общая постановка задачи дискретной оптимизации

- В некотором пространстве $\mathfrak{X}$ задано **конечное или счетное множество допустимых альтернатив** $X \subseteq \mathfrak{X}$
- На множестве X задана функция $F: X \to \Re$ – **критерий оптимизации**
- Задача – найти **оптимальную допустимую альтернативу**, то есть

$$x^* \in \operatorname{Arg}\max_{x \in X} F(x)$$

или все **множество оптимальных альтернатив**

$$X^* = \operatorname{Arg}\max_{x \in X} F(x)$$

Общая постановка задачи целочисленного программирования

- В некотором пространстве $\mathfrak{X} = Z^n$ **с помощью системы неравенств** $0 \leq g_i(x), i = 1, \dots, m$ задано **конечное или счетное** множество допустимых альтернатив $X \subseteq \mathfrak{X}$
- На множестве X задана функция $F(x)$ – **критерий оптимизации**
- Задача – найти **оптимальную допустимую альтернативу**, то есть найти, где достигается $\max_{x \in \mathfrak{X}} F(x)$ по всем x, таким, что $0 \leq g_i(x), i = 1, \dots, m$

Методы решения задач оптимизации

- «Аналитические»
  - «Компактная» форма записи алгоритма
  - Непрерывность относительно частичного порядка
- «Непрерывные»
  - Непрерывная релаксация
  - Лагранжева релаксация
- **Алгоритмические**

Классификация алгоритмических методов решения

- Классификация по трудоемкости:
  - *Полиномиальные* алгоритмы
  - *Псевдо-полиномиальные* алгоритмы
  - *Экспоненциальные* алгоритмы
  - Сверхэкспоненциальные алгоритмы

Классификация алгоритмических методов решения

- Классификация по точности
  - Точные алгоритмы
  - Приближенные (эвристические) алгоритмы
    - С гарантированными оценками качества (для относительной погрешности – класс APX) например, задача о ранце принадлежит классу APX
    - С регулируемой точностью
    - Без оценок качества

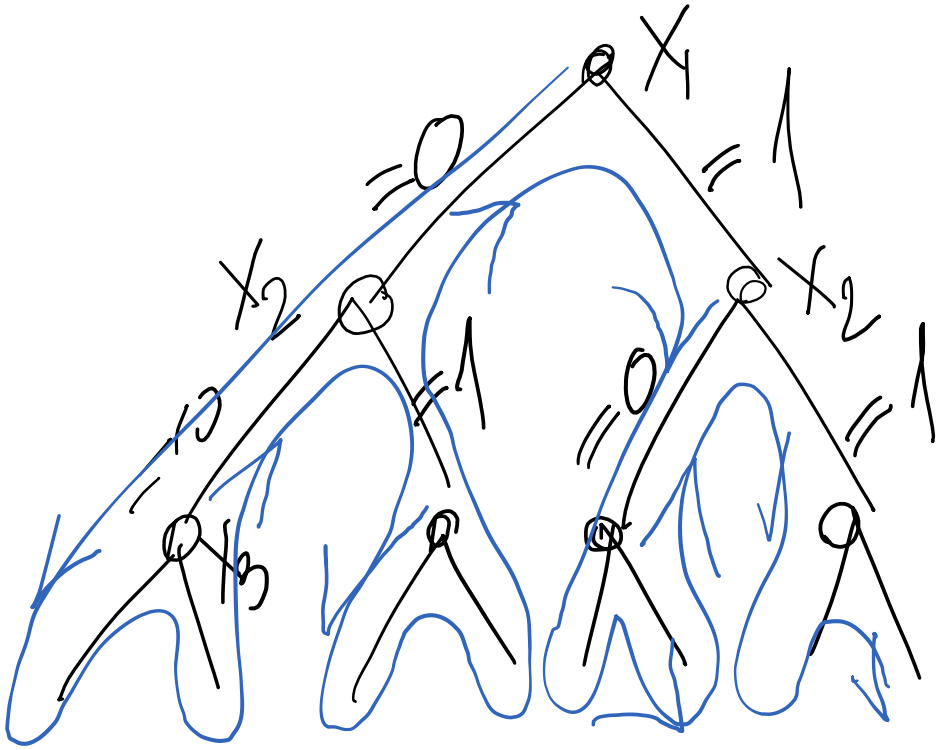
Точные алгоритмы

- Перебор
- Метод ветвей и границ
- Динамическое программирование
- Графический метод
- Дихотомическое программирование

- Идея всех точных алгоритмов – отсечение гарантированно неперспективных множеств вариантов

Полный перебор

- Универсальный алгоритм поиска (С. Бир)
- Для задачи двоичного программирования: поиск в глубину



- Поиск в ширину можно распараллелить

Метод ветвей и границ

- Необходима нижняя (для минимизации) или верхняя (для максимизации) оценка $\tilde{F}\ (X')$ решения для подмножеств X' множества *X.*
- На примере задачи максимизации:

1. Фиксируем некоторое базовое решение (рекорд) $\overline{x}$
2. Делим *X* на несколько частей (в задачах двоичного программирования обычно делится по значению переменной $x_i = \{0, 1\}$: $X_1, \dots, X_k$.
3. Вычисляем верхнюю оценку $\tilde{F}\ (X_1), \dots, \tilde{F}\ (X_k)$.
4. Если $\tilde{F}\ (X_1) < F(x)$, то решение не может быть в X_1.
5. В противном случае, если X_1 состоит из единственного элемента – решение найдено.
6. Иначе переходим на шаг 2, но уже для более узкого множества X_1.
7. Повторяем шаги 4-6 для множеств $X_2, \dots, X_k$.
8. Получается дерево ветвлений, на котором реализуется поиск в глубину с **неполным перебором** – некоторые поддеревья не рассматриваются, отбрасываются, на основе верхней оценки.

Динамическое программирование

- Динамическое программирование – раздел математического программирования, посвященный исследованию многошаговых задач принятия оптимальных решений. При этом многошаговость задачи отражает реальное протекание процесса принятия решений во времени, либо вводится в задачу искусственно за счет расчленения процесса принятия однократного решения на отдельные этапы, шаги. Цель такого представления состоит в сведении исходной задачи высокой размерности к решению на каждом шаге задачи меньшей размерности
- Для применимости динамического программирования необходимо, чтобы общий «доход» $F(x)$ за n шагов был равен сумме «доходов» от отдельных шагов. Получается не со всеми, но со многими задачами
- Последовательность допустимых решений (допустимого выбора) на отдельных шагах обычно называют *стратегией*. Требуется найти среди всех стратегий оптимальную, дающую максимум суммарного дохода
- Вводится понятие состояния, в которое система переходит после каждого шага
- Для эффективности алгоритма ДП необходимо, чтобы число (достижимых) состояний было много меньше числа вариантов (путей) перехода в это состояние
- В основе лежит принцип оптимальности Беллмана: каково бы ни были начальное состояние и первое решение, последующие решения составляют оптимальную стратегию по отношению к начальному состоянию, полученному в результате первого решения.
- Примеры:
  - Задача о кратчайшем пути (поиск в ширину/глубину)
  - Задача о ранце

Графический метод

- Модификация метода динамического программирования, основанная на том, что промежуточное решение можно вычислять не для всех состояний
- Рассмотрим на примере задачи о ранце

- Литература:
  - Лазарев А.А., Гафаров Е.Р. ТЕОРИЯ РАСПИСАНИЙ: ЗАДАЧИ И АЛГОРИТМЫ. - М.: МГУ, 2011.

Дихотомическое программирование

- Модификация графического метода
- Вместо линейной структуры («беллмановской ветви») – дерево
- Позволяет параллелить решение
- Частный случай задачи сетевого программирования

Пример – целочисленная задача о ранце

Номер	1	2	3	4
Вес c_i	2	1	3	4
Ценность p_i	3	2	4	5

R = 5

- Решить методом ветвей и границ с верхней оценкой типа затраты-эффект
- Решить методом динамического программирования
  - Уравнение Беллмана: $P(k, r)$ - максимальная ценность рюкзака размера *r* если доступны первые *k* предметов.

$$P(k, r) = \max[P(k-1, r); p_k + P(k-1, r-c_k)]$$

- Решить задачу графическим методом

Другие числа – самостоятельная работа

Номер	1	2	3	4
Вес c_i	2	3	5	7
Ценность p_i	5	7	6	3

R = 9

- Решить методом ветвей и границ с верхней оценкой типа затраты-эффект
- Решить методом динамического программирования
  - Уравнение Беллмана: максимальная ценность рюкзака размера *r* если доступны первые *k* предметов.

$$P(k,r) = \max[P(k-1,r); p_k + P(k-1, r-c_k)]$$

- Решить задачу графическим методом

Приближенные алгоритмы

- Классификация по точности
  - С гарантированными оценками качества
    - Для любой задачи независимо от исходных данных есть оценка абсолютного или относительного качества
    - APX = класс задач, для которых существует полиномиальный алгоритм фиксированного качества
  - С регулируемой точностью
    - Позволяют достигать любого качества за счет повышения времени выполнения: для любого ε >0 найдется полиномиальный алгоритм, дающий такое решение *F*, что $(F^* - F)/F^* < \varepsilon$.
    - Зависимость степени полинома от ε может быть любой
    - Задача о ранце принадлежит классу PTAS
  - Без оценок качества

Приближенные алгоритмы

– Классификация по типу оптимизации
  - детерминированные
  - вероятностные (стохастические)
  - комбинированные

Алгоритмы локальной оптимизации

- Трудно жить без понятия окрестности, широко используемого в теории непрерывной оптимизации
- Простейший алгоритм локального поиска
  - На множестве *X* вводится «топология» – семейство окрестностей $T(x) \subseteq X$ для каждого $x \in X$ Выбирается стартовая точка $x_0 \in X$ и вычисляется $F(x_0)$
  - Если среди сосед ей $x \in T(x_0)$, есть такой, для которого $F(x) < F(x_0)$ (для задачи минимизации), переходим к элементу $x_1 \in \mathrm{Argmin}_{x \in T(x_0)} F(x)$ и возвращаемся на предыдущий шаг.
  - В противном случае выдаем x_i в качестве решения.
  - Могут применяться и другие правила остановки.

Локальный поиск

- Может давать глобально оптимальное решение (правило МакНотона для минимизации времени обработки с отказами)
- Обычно дает локальный оптимум
- Условия глобального оптимума: достижимость оптимального решения по любому пути поиска
- Монотонность функции на частичном упорядочении решений позволяет сузить множество возможных решений до множества терминальных (относительно данного частичного порядка) вершин
- Предложите схему локального поиска для задачи о ранце
- Принципы выбора частичных порядков: не слишком большая «ветвистость», максимальное узкое множество терминальных вершин

Эвристические алгоритмы

- Алгоритмы, основанные на правдоподобных, но не обоснованных математически предположениях о свойствах оптимального решения задачи.
- Фактически в эвристическом алгоритме учитывается одно или несколько свойств оптимального решения, на основе которых производится сокращение перебора возможных решений.
- Обычно эвристика строится из «здравого смысла», и потому сильно зависит от задачи
- Но есть **«жадные» (greedy) алгоритмы** – заменять решение большой задачи решением последовательности локальных задач
  - Пример – эвристика «затраты-эффект» для задачи о ранце
  - Какова трудоемкость этой эвристики?

Эвристические алгоритмы

- Алгоритмы, основанные на правдоподобных, но не обоснованных математически предположениях о свойствах оптимального решения задачи.
- Фактически в эвристическом алгоритме учитывается одно или несколько свойств оптимального решения, на основе которых производится сокращение перебора возможных решений.
- Обычно эвристика строится из «здравого смысла», и потому сильно зависит от задачи
- Но есть **«жадные» (greedy) алгоритмы** – заменять решение большой задачи решением последовательности локальных задач
  - Пример – эвристика «затраты-эффект» для задачи о ранце
  - Какова трудоемкость этой эвристики?

Жадный алгоритм на основе метода ветвей и границ

- Необходима нижняя (для минимизации) или верхняя (для максимизации) оценка $F\sim(X')$ решения для любого подмножества X' множества X.
- На примере задачи минимизации:

1. Базовое решение не нужно
2. Делим X на несколько частей (в задачах двоичного программирования обычно делится по значению переменной $x_i = \{0,1\}$): $X_1, \ldots, X_k$.
3. Вычисляем нижнюю оценку $F\sim(X_1), \ldots, F\sim(X_k)$.
4. Выбираем множество X_i с минимальной нижней оценкой.
   1. Или среди всех «крайних» вершин дерева ветвлений
   2. Или среди дочерних вершин текущей вершины
5. Если X_i состоит из единственного элемента – решение найдено.
6. Иначе переходим на шаг 2, но уже для более узкого множества X_i.

- Если нижняя оценка точная, алгоритм всегда дает оптимальное решение
- Можно использовать для получения базового решения в МВГ
- Пример – метод ВиГ для задачи о ранце
- Не обязательно нижняя оценка, это может быть любое приближение решения.
- Пример – построение оптимального дерева принятия решений

Вероятностные методы

- Метод Монте-Карло
  - Группа численных методов, основанных на получении большого числа реализаций случайного процесса, который формируется таким образом, чтобы его вероятностные характеристики совпадали с аналогичными величинами решаемой задачи.
  - Для ранца
    - Если $\lambda_i = \frac{p_i}{c_i}$ - эффективность предмета, то можно отбирать предметы в ранец на основе вероятности $\frac{\lambda_i}{\sum_{j=1}^{n} \lambda_j}$.
    - При многократных повторах вероятность найти оптимальное решение будет стремиться к единице.
    - Какова трудоемкость метода?
  - Более вдумчивый подход – выбрать набор характеристик задачи, (типа нормированных эффективностей предметов в задаче о ранце), и исследовать статистическую гипотезу о виде оптимального решения в зависимости от этих характеристик (вероятность попадания предмета в ранец в зависимости от нормированной эффективности, числа предметов, объема ранца, и т.д.)
  - Большое преимущество метода – распараллеливается элементарно

Вероятностные методы

- Метаэвристические алгоритмы
  - Методы комбинирования различных эвристических решений
    - Поиск с запретами (tabu search)
    - Генетические алгоритмы
    - Метод муравьиных колоний
    - Метод имитации отжига

Метаэвристические алгоритмы

- **Поиск с запретами (*tabu search*) – модификация локального поиска.**
  Позволяет продолжить поиск, осуществляя переход к решению из окрестности, даже если значение целевой функции при новом решении хуже. Это позволяет «выскакивать» из локальных экстремумов.
  Недостаток – возможное зацикливание, когда происходит возврат к уже просмотренному ранее решению.
  Чтобы избежать зацикливания, некоторые решения или переходы от решения к решению объявляются запрещенными (табу). Для этого создается и хранится «список запретов», в который помещается информация о последних решениях или последних изменениях решений.
- В методе **«имитация отжига»** (*simulated annealing*) точка из окрестности выбирается случайно. Переход происходит всегда, когда в окрестности находится лучшее решение, и с некоторой вероятностью, если текущее решение – локальный экстремум.
  Вероятность пропорциональна параметру, т.н. *температуре*, и обратно пропорциональна величине потерь от перехода к новому решению.
  Например, $P = e^{-\Delta/T}$.
  Для сходимости метода на каждом шаге температура уменьшается (имитация остывания).

Метаэвристические алгоритмы (продолжение)

- В **генетических алгоритмах** стараются комбинировать «хорошие» решения, чтобы получить еще «лучшее». В генетике также пытаются скрещивать разные биологические виды (или особей одного вида), чтобы полученное потомство обладало полезными качествами своих родителей.
- Метод **муравьиные колонии** основан на следующем наблюдении из жизни муравейника. Муравьи пытаются отыскать пищу поближе к муравейнику. О том, где находится ближайшая пища, они сообщают друг другу следующим образом. При своем передвижении муравьи оставляют на поверхности “феромонные следы”, “запах” которых могут чувствовать другие муравьи. Очередной муравей при поиске пищи ориентируется на феромонный след других муравьев и старается с небольшими отклонениями следовать в том направлении, куда ведет большинство следов. Небольшие отклонения от общего направления позволяют строить новые маршруты и находить новую пищу, а накопление феромона позволяет двигаться в разумном (локально оптимальном) направлении.

Генетические алгоритмы

- Этапы генетического алгоритма:
  - Задать целевую функцию (функцию приспособленности, fitness) для особей популяции
  - Создать начальную популяцию
  - Цикл
    - Размножение (скрещивание)
    - Мутирование
    - Вычислить значение целевой функции для всех особей
    - Формирование нового поколения (селекция)
    - Если выполняются условия остановки, то **Конец**.

Я лично никогда не сталкивался ни с одной задачей, для решения которой генетические алгоритмы оказались бы самым подходящим средством. Более того я никогда не встречал никаких результатов вычислений, полученных посредством генетических алгоритмов, которые производили бы на меня положительное впечатление.

Стивен С. Скиена

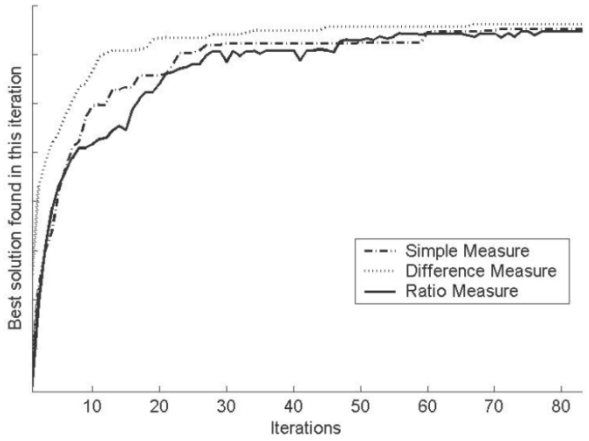
Генетический алгоритм для задачи о ранце

- A.E. Eiben, J.E. Smith. *Introduction to Evolutionary Computing.* Springer-Verlag, 2010. First Edition: Springer-Verlag, 2003
  - Хромосома – бинарный вектор вхождения предметов в ранец
  - Начальная популяция – случайная
  - Размер популяции = 500 особей
  - Функция приспособленности – суммарная ценность предметов
  - Отбор родителей – выбор лучшего из двух случайных
  - Вероятность скрещивания 70% (при отсутствии скрещивания клон + мутация
  - Скрещивание – разрывом хромосом в одной точке (2 родителя, 2 потомка), недопустимые потомки отбрасываются
  - Мутация – инверсия каждого бита.
  - Вероятность инверсии – 1/n, в среднем – одна мутация на одного потомка
  - Выживают только дети – предыдущее поколение полностью очищается
  - Условие остановки – нет улучшения на протяжении 25 поколений
  - Всегда запоминаем лучшее среди всех найденных решений

Как работает? «Awesome. It took a little longer than the dynamic programming solution (actually ~465 times longer).»

Метод муравьиных колоний для задачи о ранце

- В основном используется для обобщений задачи (многомерный ранец, стохастический ранец)
- 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 $$\tau_{ij}(t) = \tau_{ij}(t) \times (1 - \rho).$$
- См. презентацию



Комбинированные эвристики: Генетический локальный поиск

Шаг 1. Построить начальную популяцию.
Шаг 2. Пока не выполнен критерий остановки, делать следующее:
Шаг 2.1. Выбрать два решения из популяции.
Шаг 2.2. Построить по ним новое решение.
Шаг 2.3. Применить к нему алгоритм локального улучшения.
Шаг 2.4. Добавить новый локальный оптимум в популяцию и удалить из нее наихудший локальный оптимум.
Шаг 3. Предъявить лучшее найденное решение.

Точно так же локальный поиск комбинируется с любой другой эвристикой, дающей начальные точки

Приближенные алгоритмы гарантированного качества

Задача поиска приближенного решения

- Пусть x_0 – точка минимума $f(x)$. Необходимо найти такой x^*, что

$$f(x^*) - f(x^0) \le \varepsilon \qquad (1)$$

Или $$\frac{f(x^*) - f(x^0)}{f(x^0)} \le \varepsilon, \quad f(x^0) \ne 0, \quad \varepsilon > 0. \qquad (2)$$

Задача (2) обычно проще, чем (1)

- А как записать качество алгоритма максимизации?

Гарантированное качество алгоритма Данцига (метода затраты-эффект) для задачи о ранце

- $f(x) = \sum_{i=1}^{n} x_i \cdot p_i \to max$ при $\sum_{i=1}^{n} x_i \cdot c_i \leq R$.
- Пусть $R = c_1 \cdot k, k > 1$. тогда $k \leq k_0$, где $k_0 < \sum c_i / c_1$. Пусть $\lambda_i := p_i / c_i$ и $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$.
- Обозначим $\epsilon_0 = 1 - 1/k_0$
- Теорема: алгоритм Данцига имеет точность не хуже ϵ_0. То есть если алгоритм дает решение x^*, а x_0 – оптимальное решение , то $\frac{f(x_0) - f(x)}{f(x_0)} \leq \epsilon_0$
- Итого, оценка, не зависящая от параметров задачи (от k_0) ϵ_0= ? (скажите, сколько?)

Гарантированное качество алгоритмов решения метрической задачи коммивояжера

- Алгоритм двойного обхода остового дерева
  - Задача об остовом дереве - пример простой задачи дискретной оптимизации. Эффективно решается жадным алгоритмом
  - Маршрут коммивояжера без одной дуги - это тоже остовное дерево
  - Двойной обход минимального остового дерева заменяется маршрутом коммивояжера не большей длины.
  - Какова оценка гарантированного качества ϵ?
- Алгоритм Кристофидеса с паросочетаниям (основан на сведении к эйлеровому циклу в частичном графе)
  - Находим кратчайший остов
  - Ищем оптимальных взвешенных паросочетаний на множестве вершин нечетной степени (дополняем остов до эйлерова графа)
  - Находим в дополненном остове эйлеров цикл
  - Стягиваем в гамильтонов цикл не большей длины
  - $\epsilon = 1/2$
- Наилучший известный алгоритм для расстояний из {1, 2} имеет оценку 1/8

Возможность полиномиальной аппроксимации задачи о ранце

- Ю. Ю. Финкельштейн, ε-подход к многомерной задаче о ранце: полиномиальный рост дерева ветвления, *Ж. вычисл. матем. и матем. физ.*, **17**:4 (1977), 1040–1042
- Задача о многомерном ранце

$$f(x) = \sum_{j=1}^{n} c_j x_j \to \max, \tag{5.4.1}$$

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i, \quad i = 1, 2, \ldots, m, \tag{5.4.2}$$

$$x_j \in \{0; 1\}, \quad j = 1, 2, \ldots, n, \tag{5.4.3}$$

$$c_j > 0, \quad j = 1, 2, \ldots, n, \tag{5.4.4}$$

$$b_i > 0, \quad i = 1, 2, \ldots, m, \tag{5.4.5}$$

$$0 \le a_{ij} \le b_i, \quad i = 1, 2, \ldots, m, \quad j = 1, 2, \ldots, n, \tag{5.4.6}$$

$$\sum_{j=1}^{n} a_{ij} > b_i, \quad i = 1, 2, \ldots, m, \tag{5.4.7}$$

$$c_1 \ge c_2 \ge \cdots \ge c_n. \tag{5.4.8}$$

Возможность полиномиальной аппроксимации задачи о ранце

- Ю. Ю. Финкельштейн, ε-подход к многомерной задаче о ранце: полиномиальный рост дерева ветвления, *Ж. вычисл. матем. и матем. физ.*, **17**:4 (1977), 1040–1042
- Алгоритм ВиГ для задачи о (многомерном) ранце с правилом отбора (f~(Xi)-f(x))/f(x)<=ε и правилом ветвления по первой по порядку нецелой компоненте решения линейной релаксации дает ε-оптимальное решение

ОПРЕДЕЛЕНИЕ. Переменная x_j называется *ε-существенной*, если в процессе работы ε-оптимального алгоритма по ней сделано хотя бы одно ветвление.

Пусть $x' = (1; 0; \ldots; 0)$, либо x' — любое допустимое решение такое, что $f(x') \geq c_1$.

ТЕОРЕМА 5.4. *Для числа ε-существенных переменных имеет место оценка*

$$n_\varepsilon \leq \frac{m}{\gamma\varepsilon}; \qquad (5.4.13)$$

здесь $\gamma = f(x') \Big/ \sum_{j=1}^{n} c_j$.

СЛЕДСТВИЕ. *Число итераций (ветвлений) J_ε при решении задачи (5.4.1)–(5.4.8) по ε-оптимальному алгоритму A_ε оценивается следующим образом:*

$$J_\varepsilon \leq 2^{m/\gamma\varepsilon}. \qquad (5.4.14)$$

5.4.3. Замечания о полиномиальной трудоемкости алгоритма. В дальнейшем Ю. Ю. Финкельштейном было показано, что ε-оптимальный алгоритм A_ε при фиксированных m и ε имеет полиномиальную по n оценку числа ветвлений *). Степень многочлена была оценена и оказалось, что она зависит от $1/\varepsilon$.