Лекция 2. Задачи дискретной оптимизации и их сложность

Общая постановка задачи дискретной оптимизации. Классификация методов решения. Сложность задач дискретной оптимизации. Классы *P* и *NP*. Задача о ранце.

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 $\mathfrak{x} = Zn$ **с помощью системы неравенств** $0 \leq gi(x), i = 1, \dots, m$ 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 $0 \leq gi(x), i = 1, \dots, m$.

Пример: задача о ранце

- Имеется множество предметов $N = \{1, \dots, n\}$ с весами $c_i, i = 1, \dots, n$ и ценностями $p_i,\ i = 1, \dots, n$, и рюкзак емкости *R*.
- Необходимо выбрать подмножество предметов *N'* суммарным весом не более *R*, имеющее максимальную ценность.
- $\sum_{i=1}^{n} x_i \cdot p_i \to max$ при $\sum_{i=1}^{n} x_i \cdot c_i \leq R$, $x_i \in \{0,1\}$
- На протяжении всего курса будем в основном заниматься решением модификаций задачи о ранце.

Методы решения задач дискретной оптимизации

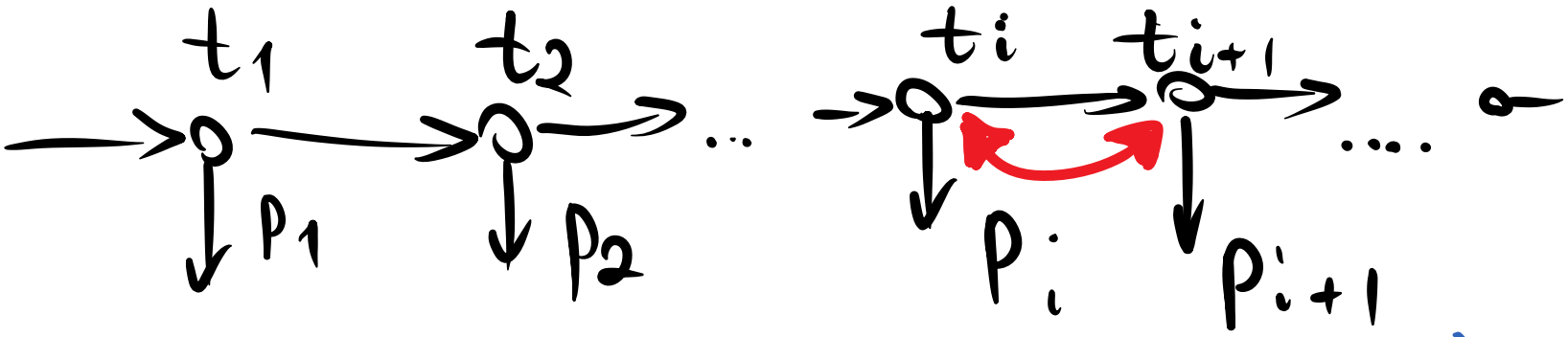
- «Аналитические»
- «Непрерывные» - релаксации для получения верхних и нижних оценок
  - Непрерывная релаксация (пример – задача о ранце)
  - Лагранжева релаксация
  - *Сетевое программирование* (рассмотрим отдельно позже)
- Алгоритмические

Аналитические методы решения задач оптимизации

- Аналитическое решение как компактная запись алгоритма
  - Пример с задачей отбраковки – правило Мак-Нотона
- Изучение свойств задачи
  - Сужение множества альтернатив
  - Избавление от неудобных ограничений
  - Выявление легко решаемых частных случаев
- Основная идея - монотонность относительно частичного порядка

Аналитическое решение как компактная запись алгоритма

- Задача отбраковки – правило Мак-Нотона



$$t_i + (1 - p_i) t_{i+1} \leqslant t_{i+1} + (1 - p_{i+1}) t_i$$

$$\Downarrow$$

$$\frac{t_i}{p_i} \leqslant \frac{t_{i+1}}{p_{i+1}}$$

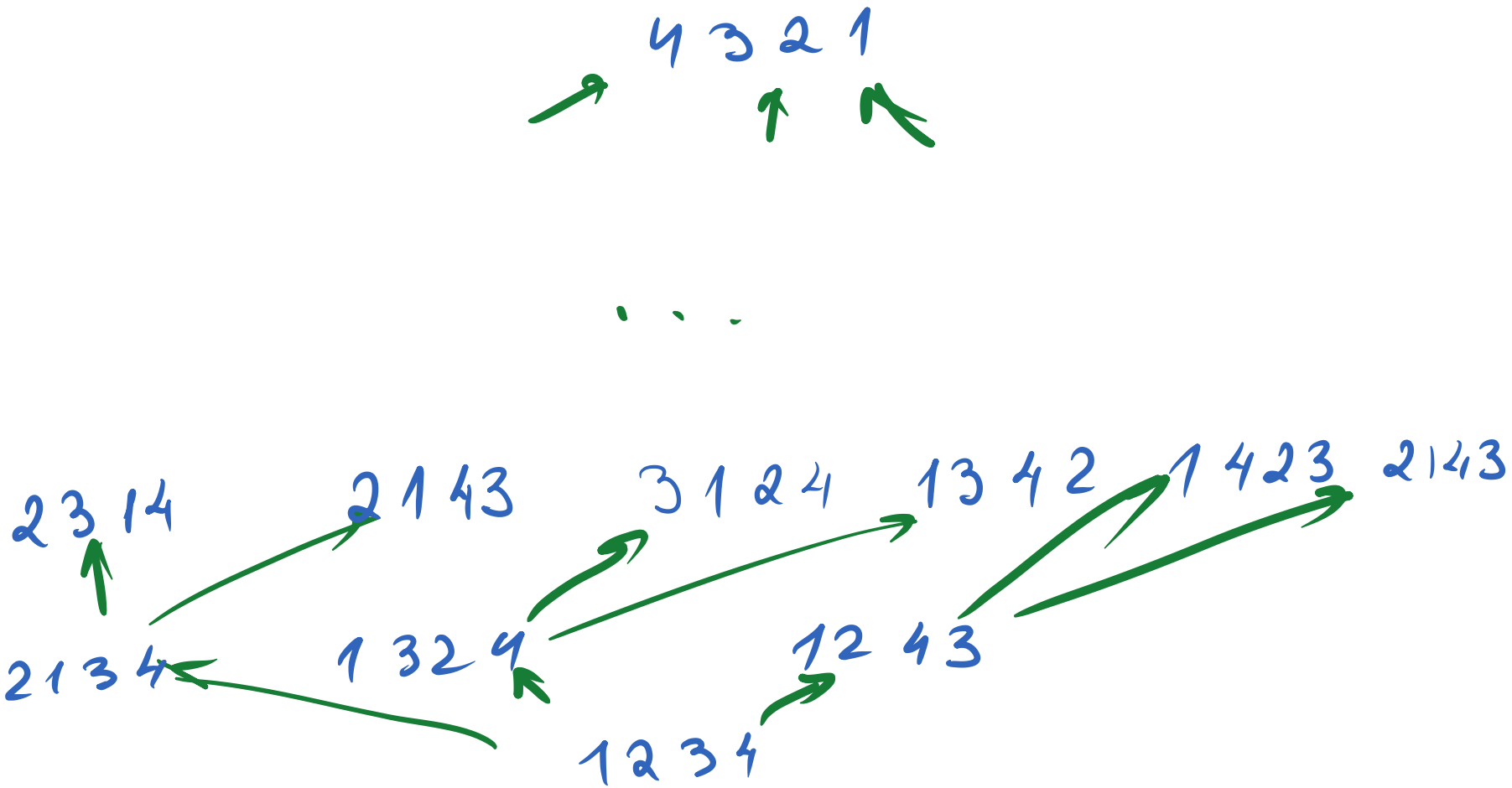
Аналитические методы решения задач оптимизации

- Изучение свойств задачи
  - Сужение множества альтернатив
  - Избавление от неудобных ограничений
  - Выявление легко решаемых частных случаев

- Основная идея - монотонность относительно частичного порядка

(какой частичный порядок использован в задаче отбраковки?)

Монотонность функции относительно частичного порядка



«Непрерывные» методы решения задач дискретной оптимизации

- Релаксация задачи о ранце
- Имеется множество предметов $N = \{1, \dots, n\}$ с весами $c_i, i = 1, \dots, n$ и ценностями $p_i, i = 1, \dots, n$, и рюкзак емкости R.
- $\sum_{i=1}^{n} x_i \cdot p_i \to max$ при $\sum_{i=1}^{n} x_i \cdot c_i \leq R$. $x_i \in [0,1]$.
- Задача линейного программирования

«Непрерывные» методы решения задач дискретной оптимизации

- Лагранжева релаксация
- Задача:

P: $v_P = \min cx$ при $Ax \geq b, x \in \{0,1\}$

- Релаксация

LR: $v_{LR}(\lambda) = \min cx + \lambda(b - Ax)$ при $x \in \{0,1\}$

- Двойственная задача

$$\max v_{LR}(\lambda), \lambda \geq 0$$

- Метод ее решения – субградиентная оптимизация

$\lambda^{k+1} = \lambda^k + t_k (A\, x^k - b)$, где x^k – решение задачи минимизации *LR* при множителе Лагранжа, равном λ^k.

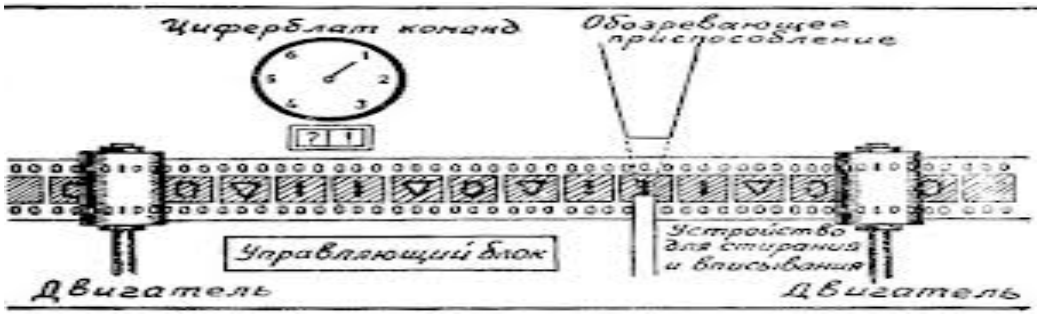
- В общем случае $v_P \geq v_{LR}$.
- Литература:

Поляк Б.Т. Минимизация негладких функционалов, ИСВМ и МФ т9 (1969), №3, с. 509-521.

Алгоритмы

Алан Матисон
Тьюринг
(1912 - 1954)

- – Инструкция для компьютера
  - – Машина Тьюринга (детерминированная одноленточная)
    - Неограниченная в обе стороны *лента*, разделённая на ячейки
    - *головка записи-чтения*, способная находиться в одном из множества *состояний*.
    - Головка может перемещаться по ленте, читать и записывать в ячейки символы конечного алфавита.
    - П*равила перехода* предписывают машине, в зависимости от текущего состояния и наблюдаемого в текущей клетке символа, записать в эту клетку новый символ, перейти в новое состояние и переместиться на одну клетку влево или вправо.
    - При переходе машины Тьюринга в *терминальное состояние* алгоритм останавливается.
  - – Все ли равно, какой компьютер?
    - RAM
    - Квантовые компьютеры



Как измерить сложность задачи?

- Есть задачи настолько сложные, что вообще не существует алгоритма для их решения!
  - Проблема остановки машины Тьюринга (останавливается ли алгоритм на произвольно заданном входе?)
  - Десятая проблема Гильберта (разрешимость в целых числах полиномиальных уравнений)
  - …
- Но как измерить сложность **разрешимых** задач?

Рациональные схемы кодировки

(а) *Код примера задачи должен быть компактным и не должен содержать необязательную информацию или символы.*

(б) *Числа из примера задачи должны быть представлены в двоичной, либо какой-нибудь иной системе счисления с основанием, отличным от единицы.*

(в) *Схема кодирования должна обладать свойством декодируемости,* т.е. для каждого параметра задачи должен существовать полиномиальный алгоритм, способный выделить код этого параметра из кода любого примера задачи.

(г) *Схема кодирования должна обеспечивать однородность при кодировании различных примеров одной и той же задачи.* Однородность означает, что схема кодирования должна обеспечивать для любых двух различных примеров задачи получение кодов, "близких" по длине, если эти примеры содержат близкие по величине числа и количество параметров в обоих примерах приблизительно одно и то же.

Временная сложность алгоритма

- Класс P полиномиально разрешимых задач
  - Если существует алгоритм решения задачи, трудоемкость которого в худшем случае равна $O(n^k)$, где k – некоторая константа, не зависящая от длины n кода задачи, то говорят, что задача **полиномиально разрешима**
- Класс QP квазиполиномиально разрешимых задач
  - Если существует алгоритм, трудоемкость которого полиномиально зависит не только от размерности задачи, но и от значений числовых параметров примера, то говорят, что алгоритм **псевдополиномиальный**
- Класс **EXP** задач для которых существует алгоритм сложности порядка $O(2^{n^k})$
- Полиномиальная сводимость задач

Задачи оптимизации и распознавания

- Задача распознавания, соответствующая задаче оптимизации $P=\min_{x\in\Omega} F(x)$ дает ответ на вопрос «верно ли, что $P \geq P_0$?».
- Сложность алгоритма (задачи) оптимизации
  - Теорема: если критерий оптимизации принимает не более N значений, то существует процедура решения (какая?) оптимизационной задачи за $\log_2 N$ шагов решения задачи распознавания
  - Следствие: если $\log_2 N$ растет не быстрее полинома от длины входа, задачи оптимизации и распознавания полиномиально сводимы
  - Пример, когда $\log_2 N$ растет быстрее полинома от длины входа: минимизировать произведение $2^n/n$ различных решений задачи о ранце

Класс NP

- Теория NP-полных задач для удобства строится только для задач распознавания
- Класс NP
  - Класс ***NP*** содержит все полиноминально проверяемые задачи распознавания. В них для каждого примера с ответом “ДА” оракул выдаст решение (догадку), такое что:
    - длина решения полиномиально ограничена длиной входных данных , и
    - решение может быть проверено за полиномиальное время от размерности задачи.
- Теорема. $NP \subseteq EXP$.

Доказательство: Пусть проверка догадки имеет сложность n^k. Значит, длина любой догадки не более n^k. Тогда при алфавите размера L догадок не может быть больше L^{n^k}. Тогда сложность алгоритма, проверяющего все догадки не будет больше $n^k L^{n^k}$, а эта функция доминируется некоторой экспонентой.

NP-полные и NP-трудные задачи

- Задача NP-трудная, если к ней полиномиально сводима любая другая задача класса NP
- NP-трудная задача называется NP-полной, если вдобавок она принадлежит NP (самые сложные задачи класса *NP*). NP-полные задачи существуют (Кук, задача о выполнимости)
- Задача *B* называется *NP*-трудной в сильном смысле, если для нее не существует псевдо-полиномиальный алгоритм решения (в предположении $P \neq NP$).
- NP-полные задачи дискретного программирования имеют в худшем случае экспоненциальную сложность $O(2^{n^k})$.

Как доказать NP-полноту

- Воспользоваться транзитивностью полиномиальной сводимости
  - Доказать NP-полноту хотя бы одной задачи
  - Свести ее к нашей задаче
- Теорема Кука – доказательство NP-полноты задачи выполнимости (задана булева формула вида $f(x) = x_1 \cap (x_2 \cup \neg x_1) \cap \ldots \cap (x_2 \cup x_i \cup \neg x_n)$, $x_i \in \{true, false\}$. Есть ли комбинация переменных, при которых $f(x) = true$?)
- Выполнимость -> 3-выполнимость -> 3-сочетание -> разбиение -> ранец

P = NP?

- Вопрос на миллион.

- Задача распознавания *Q* – дополнение задачи распознавания *P* – если множество примеров задачи *P*, имеющих ответ «Да», равно множеству примеров задачи *Q*, не имеющих ответ «Да».
- P=co-P Докажите!
- NP =co-NP?

Пример анализа сложности задачи

- Задача о ранце: NP-полная, но принадлежит классу QP.

Другие понятия сложности

- Важна не только временная сложность. Для работы алгоритмов требуется еще и память. Здесь отдельная теория.
  - Как соотносятся между собой классы задач, требующих для своего решения логарифмической, линейной, полиномиальной, экспоненциальной, сверхэкспоненциальной памяти?
- Еще есть понятие схемной сложности – длина алгоритма

- Дополнительная литература:
  - *Гэри М., Джонсон Д.* Вычислительные машины и труднорешаемые задачи. Пер. с англ. М.: Мир, 1982. 416 с.
  - Пападимитриу Х., Стайглиц К. Комбинаторная оптимизация. М.: Мир, 1985.