Дискретные задачи теории принятия решений

II часть

Механико-математический факультет

4 курс, 2 семестр

Лектор: **Алексеева Екатерина Вячеславовна**

Анализ алгоритмов, сложность задач. Теория NP-полноты.

Лекция 1

Три задачи

Задача Π – это отношение $\Pi \subseteq I \times S$, где I – множество примеров, S – множество решений задачи.

Задача распознавания (задача принятия решения)

В качестве отношения Π выступает функция $f : I \to S$, где $S = \{0,1\}$ или $S = \{да, нет\}$

Задача поиска

Для любого примера $x \in I$ найти решение $y \in S$, такое что $(x, y) \in \Pi$

Оптимизационная задача

Для данного примера $x \in I$ требуется найти «наилучшее» решение y^* среди всех $y \in S \mid (x, y) \in \Pi$

Вычислительный анализ сложности

Input: неотрицательные целые числа x, y
Output: неотрицательное целое число $r=x^y$

```
Begin
    r := 1;
    while y ≠ 0 do
    begin
        r := r * x;
        y := y – 1;
        Return r;
    end;
End.
```

$T_1=2+3y$

$T_2=aylogy+by^2logx(logy+loglogx)+c$

Размер входных данных

Определить является ли $x \in Z^+$ простым числом?

Алгоритм проверяет все возможные делители d,
$1 \le d \le \sqrt{x}$

Представим число x в виде бинарной строки длиной
$n = |x| \approx \log x$, тогда $T \approx 2^{n/2}$

Асимптотический анализ алгоритмов

Пусть $f, g : \mathrm{N} \to \mathrm{N}$ говорят, что

$f(n) = O(g(n))$, если существуют константы c, a, n_0 |

$\forall\ n \geq n_0\ \ f(n) \leq cg(n) + a$

$f(n) = \Omega(g(n))$, если существуют константы c, a, n_0 |

$\forall\ n \geq n_0\ \ f(n) \geq cg(n) + a$

$f(n) = \Theta(g(n))$, если $f(n) = O(g(n))$ и $f(n) = \Omega(g(n))$

Асимптотический анализ алгоритмов

Пусть $\hat{T}_A(x)$ время работы алгоритма на входе x, тогда время работы алгоритма в худшем случае

$$T_A(x) = \max\{\hat{T}_A(x) \mid \forall x : |x| \le n\}$$

Пусть $\hat{S}_A(x)$ размер памяти для реализации алгоритма на входе x, тогда размер памяти для реализации алгоритма в худшем случае $S_A(x) = \max\{\hat{S}_A(x) \mid \forall x : |x| \le n\}$

Асимптотический анализ алгоритмов

Алгоритм A имеет сложность $O(g(n))$ ***(оценка сверху)***,
если $T_A(x)=O(g(n))$

Алгоритм A имеет сложность $\Omega(g(n))$ ***(оценка снизу)***,
если $T_A(x)=\Omega(g(n))$

Алгоритм A имеет сложность $\Theta(g(n))$, если $T_A(x)=\Theta(g(n))$

Сложность задач

Задача $П$ имеет *верхнюю оценку сложности* $O(g(n))$, если существует алгоритм решения $П$ временная сложность которого $O(g(n))$.

Задача $П$ имеет *нижнюю оценку сложности* $\Omega(g(n))$, если временная сложность любого алгоритма для решения $П$ равна $\Omega(g(n))$.

Сложность задачи $П$ - $\Theta(g(n))$, если её верхняя оценка $O(g(n))$ и нижняя оценка $\Omega(g(n))$.

Задача распознавания

Задача $П$ называется *задачей распознавания*, если множество $I_П$ всех примеров задачи разбивается на два подмножества примеров $Y_П$ с ответами «да» и $N_П$ с ответами «нет». Задача заключается в том, чтобы для любого примера $x \in I_П$ проверить $x \in Y_П$?

$D_П$ - множество входов не соответствующих примеру $П$

Задача распознавания $П$ *решается алгоритмом* A, если алгоритм заканчивает свою работу на любом примере $x \in I_{П}$ и выдает ответ «да», тогда и только тогда, когда $x \in Y_{П}$.

Классы сложности задач

Класс задач решаемых за полиномиальное время

$$P = \bigcup_{k=0}^{\infty} Time(n^k)$$

Класс задач решаемых с полиномиальным размером памяти $PSPACE = \bigcup_{k=0}^{\infty} Space(n^k)$

Класс задач решаемых за экспоненциальное время

$$EXPTIME = \bigcup_{k=0}^{\infty} Time(2^{n^k})$$

$$P \subseteq PSPACE \subseteq EXPTIME,\ P \subset EXPTIME$$

Открытые вопросы: $P \subset PSPACE \subset EXPTIME$

Задача выполнимости

Дано: конъюнктивная нормальная форма F, множество булевых переменных V, значения булевых переменных, функция $f : V \to \{true, false\}$

Вопрос: выполнима ли формула на заданном наборе значений переменных?

Задача выполнимости

Дано: конъюнктивная нормальная форма F, множество булевых переменных V

Вопрос: выполнима ли формула?

Задача выполнимости булевой формулы с квантерами

Дано: конъюнктивная нормальная форма F, множество булевых переменных V

Вопрос: выполнима ли формула

$$\exists v_1 \forall v_2 \exists v_3 \ldots Q v_n F(v_1, v_2, \ldots, v_n),$$

где $Q = \exists$, если n нечетное, $Q = \forall$, если n четное?

Недетерминированный алгоритм

Недетерминированный алгоритм A решает задачу П, если на любом примере $x \in I_{П}$ алгоритм A заканчивает свою работу на любом возможном «удостоверении» и $x \in Y_{П}$, тогда и только тогда, когда существует по крайней мере одно «удостоверение», которое приводит алгоритм к ответу «да».

Недетерминированный алгоритм A решает задачу П с временной сложностью t(n), если на любом примере $x \in I_{П}$ с $|x|=n$, алгоритм A заканчивает свою работу на любом возможном «удостоверении» и $x \in Y_{П}$, тогда и только тогда, когда существует по крайней мере одно «удостоверение», которое приводит алгоритм к ответу «да» за время $t(n)$.

Для произвольной функции $f(n)$ пусть *NTIME(f(n))* множество задач распознавания, которые решаются недетерминированным алгоритмом за время $O(f(n))$.

$$NP = \bigcup_{k=0}^{\infty} NTIME(n^k)$$

$$P \subseteq NP \subset PSPACE$$

Задача распознавания $П^c$ является *дополнением* к задаче $П$, если $I_{П^c} = I_П$, $Y_{П^c} = N_П$ и $N_{П^c} = Y_П$.

Класс co-NP – это класс всех задач распознавания, являющихся дополнениями задач из *NP* называется.

Дано: конъюнктивная нормальная форма F, множество V булевых переменных

Вопрос: правда ли, что не существует значений переменных, что выполнима формула?

Теорема: если Π - задача из класса P , то Π^c также входит в класс P.

Доказательство. Т.к. $\Pi \in P$, то существует алгоритм, который решает задачу Π. Полиномиальный алгоритм для решения Π^c это тот же алгоритм с заменой «да» на «нет».

Теорема: если дополнение некоторой *NP*-полной задачи принадлежит классу *NP*, то *NP=co-NP*.

Доказательство. Пусть $\overline{C}$ дополнение некоторой *NP*-полной задачи C принадлежит *NP*, покажем, что тогда дополнение $П^c$ любой задачи $П$ из *NP* также принадлежит *NP*.

Т.к. C *NP*-полна, то $П$ полиномиально преобразуется нее. Это преобразование является также полиномиальным преобразованием $П^c$ в задачу $\overline{C}$. Поэтому можно представить короткое удостоверение для любого примера задачи $П^c$ с ответом «да»: оно состоит

из 1. списка операций указанного полиномиального преобразования, дающего в результате пример задачи $\overline{C}$ с ответом «да»

2. удостоверение для этого примера задачи $\overline{C}$. Такое удостоверение существует, т.к. $\overline{C} \in NP$.

Полное удостоверение полиномиально по длине, т.к. по предположению рассматриваемое преобразование полиномиально.

$\Rightarrow \Pi^c \in NP$. Т.к. в качестве Π была взята произвольная задача из *NP*, то *NP=co-NP*.

Сводимость задач

Задача распознавания $П_1$ *сводится по Карпу* к задаче $П_2$ ($П_1 \leq_m П_2$), если существует алгоритм R, который любой пример $x \in I_{П_1}$ переводит в пример $y \in I_{П_2}$, такой что $x \in Y_{П_1}$ тогда и только тогда, когда $y \in Y_{П_2}$.

Если $П_1 \leq_m П_2$ и $П_2 \leq_m П_1$, то *задачи эквиваленты* по Карпу ($П_1 \equiv_m П_2$).

Сводимость по Карпу *полиномиальна* ($П_1 \leq_m^p П_2$), тогда и только тогда, когда R - полиномиальный алгоритм.

$$\Pi_2 \in P \Rightarrow \Pi_1 \in P.\ \Pi_1 \notin P \Rightarrow\ \Pi_2 \notin P$$

Задача {0,1}-линейного программирования

Дано: множество булевых переменных $Z = \{z_1,...,z_n\}$,

множество I линейных неравенств на Z.

Существует ли назначения набор значений переменных, такой что все неравенства верные?

Упражнение.

Построить сведение

Задача выполнимости $\leq_m^p$ $\{0,1\}$-линейного программирования

Пусть Π_1 - задача вычисления функции $g : I_{\Pi_1} \to S_{\Pi_1}$. Π_1 *сводится по Тьюрингу* к задаче Π_2 ($\Pi_1 \leq_T \Pi_2$), если существует алгоритм R, который решает Π_1 обращаясь к оракулу для Π_2.

Если $\Pi_1 \leq_T \Pi_2$ и $\Pi_2 \leq_T \Pi_1$, то *задачи эквиваленты по Тьюрингу* ($\Pi_1 \equiv_T \Pi_2$).

Пусть задача Π заключается в вычислении функции $f : I_\Pi \to S_\Pi$. *Оракул* – это некое абстрактное устройство, которое для любого $x \in I_\Pi$ сообщает значение функции $f(x) \in S_\Pi$.

Задача распознавания Π из класса сложности C является полной задачей в этом классе или *C-полной*, относительно некоторой сводимости $\leq_r$, если для любой задачи $\Pi_1 \in C$, $\Pi_1 \leq_r \Pi$.

Класс сложности C – *замкнут* относительно некоторой сводимости $\leq_r$, если $\Pi_1 \leq_r \Pi_2$, из того, что $\Pi_2 \in C$ следует, что $\Pi_1 \in C$.

NP-полные задачи

Задача распознавания Π является *NP-полной задачей*, если она полная в классе *NP*, относительно сводимости $\leq_m^p$, т.е. если для любой задачи $\Pi_1 \in NP$, $\Pi_1 \leq_m^p \Pi$.

Оптимизационные задачи

Задача оптимизации – это $(I_{\Pi}, Sol_{\Pi}, m_{\Pi}, goal_{\Pi})$, где

I_{Π} - множество примеров задачи Π;

Sol_{Π} - функция, сопоставляющая каждому примеру

$x \in I_{\Pi}$ множество допустимых решений x;

m_{Π} - функция, определенная на паре (x, y), таких что $x \in I_{\Pi}$, $y \in Sol_{\Pi(x)}$. Для каждой пары $m_{\Pi}(x, y)$ вычисляет целое положительное число – значение на допустимом решении y;

$goal_{\Pi} \in \{\max, \min\}$.

Каждая задача оптимизации имеет соответствующую задачу распознавания. И она не сложнее оптимизационной.

Задача оптимизации $П = (I_П, Sol_П, m_П, goal_П) \in NPO$, если

- множество примеров $I_П$ распознаваемо за полиномиальное время;
- существует полином q, такой что для заданного примера $x \in I_П$, для любого y, $|y| \leq q(|x|)$ за полиномиальное время распознается $y \in Sol_{П(x)}$;
- $m_П$ вычисляется за полиномиальное время.

PO и NPO задачи

Теорема. Для любой оптимизационной задачи Π из класса NPO, соответствующая задача распознавания Π_D принадлежит классу NP.

Доказательство. Пусть Π - задача на максимум. Для $x \in I$ и целого числа K, мы можем решить Π_D следующим недетерминированным алгоритмом. Т.к. $\Pi \in NPO$, то любая строка y, $|y| \leq q(|x|)$ за время $q(x)$ отгадывается и за полиномиальное время проверяется $y \in Sol(x)$. Если да, то $m(x, y)$ вычисляется за полиномиальное время. И если $m(x, y) \geq K$, то ответ «Да», иначе (если y недопустимое решение или $m(x, y) < K$), то ответ «Нет».

PO и NPO задачи

Оптимизационная задача $\Pi \in PO$, если $\Pi \in NPO$ и существует полиномиально-вычислимый алгоритм A, такой что для любого $x \in I$ находит оптимальное решение $y \in Sol^*(x)$ и значение $m^*(x)$.

Задача о поиске кратчайшего пути между двумя вершинами в графе.

NP-трудные оптимизационные задачи

Оптимизационная задача Π называется *NP-трудной*, если любая задача принятия решения $\Pi_D \in NP$, $\Pi_D \leq_T^P \Pi$, т.е. Π_D может быть решена за полиномиальное время алгоритмом с оракулом: для любого $x \in I_\Pi$ находит оптимальное решение $y^*(x)$ и значение $m_\Pi^*(x)$.

Утверждение.
Если $P \neq NP$, то $PO \neq NPO$.

Литература и ссылки

Первоисточники по теории NP-полноты

Р. М., Карп Сводимость комбинаторных проблем. Кибернетический сборник, новая серия, вып. 12 – М.: Мир, 1975, стр. 16-38

Научно-популярная книга с понятным изложением материала

Х., Пападимитриу, К. Стайглиц. Комбинаторная оптимизация. Алгоритмы и сложность. – М. Мир, 1985

Энциклопедия по теории NP-полноты

М., Гэри, Д., Джонсон Вычислительные машины и труднорешаемые задачи. М. Мир, 1982