Задачи комбинаторной оптимизации, приближенные алгоритмы и классы аппроксимации

Ильев В.П.

Омский государственный университет

Введение

Будем рассматривать *комбинаторные оптимизационные задачи*, в которых требуется выбрать наилучшее решение из конечного множества.

Нас будут интересовать вопросы *теории сложности* этих задач:

- *вычислительная сложность* оптимизационных задач, т.е. сложность поиска оптимальных решений,
- *сложность аппроксимации* оптимизационных задач, т.е. сложность поиска приближенных решений с априорными гарантированными оценками точности.

Теория сложности

Изначально теория сложности вычислений была построена для *задач распознавания свойств (decision problems)*. Такие задачи имеют только два возможных решения: «да» и «нет».

В рамках этой теории был сформулирован вопрос

$$\mathcal{P} = \mathcal{NP} \ ?$$

Гипотеза (1971). $\mathcal{P} \neq \mathcal{NP}$.

Массовая и индивидуальная задача

Массовая задача (или просто *задача*) – класс однотипных задач.

Она содержит несколько *параметров*, конкретные значения которых не заданы. Массовая задача определяется:

- общим списком всех её параметров;
- формулировкой свойств, которыми должно обладать решение (т.е. искомый объект, напр., подграф заданного графа).

Индивидуальная задача (или *пример*) получается из массовой, если всем параметрам присвоены конкретные значения.

Типичная массовая задача – *задача коммивояжера (KB).*

Задачи распознавания

(Массовая) задача распознавания P может быть охарактеризована парой множеств $< Z_P, Y_P >$, где

Z_P – множество всех *индивидуальных задач (примеров)*,

Y_P – множество всех *примеров с ответом «да»*.

Требуется $\forall I \in Z_P$ выяснить, принадлежит ли I множеству Y_P.

Пример. Задача о выполнимости КНФ.

Под *размером* задачи P будем понимать один или несколько её параметров, характеризующих объём входных данных.

Размер часто определяется неформально: напр., $|V|$ для графа $G = (V, E)$.

Алгоритм

Наиболее известная формализация понятия «алгоритм» – *детерминированная машина Тьюринга (ДМТ)*.

Алгоритм A *решает* задачу распознавания P, если A останавливается для всех $I \in Z_P$ и возвращает ответ «да» $\Longleftrightarrow I \in Y_P$.

Более формально: Пусть Z_P^* (Y_P^*) – множество слов в некотором алфавите, являющихся кодами всех примеров задачи распознавания P (примеров с ответом «да»).

Алгоритм A *решает* задачу распознавания P, если соответствующая ДМТ-программа останавливается на всех входах $x \in Z_P^*$ и *распознает* множество Y_P^*, т.е. *принимает* $x \Longleftrightarrow x \in Y_P^*$.

Полиномиальные алгоритмы

Время работы алгоритма A при решении примера I можно оценивать числом $t_A(I)$ его шагов до остановки.

Трудоёмкость или *временная сложность* алгоритма A – это функция $T_A(n)$, которая каждому $n \in \mathbb{N}$ ставит в соответствие максимальное время работы алгоритма по всем индивидуальным задачам $I \in Z_P$ размера n:

$$T_A(n) = \max \{t_A(I) \ : \ |I| = n\}.$$

Алгоритм A называется *полиномиальным*, если существует такой полином g, что $T_A(n) \leqslant g(n)$ для всех $n \in \mathbb{N}$.

Класс $\mathcal{P}$

Класс $\mathcal{P}$ – это класс всех задач распознавания, разрешимых полиномиальными алгоритмами.

Другими словами, *задача распознавания P принадлежит классу $\mathcal{P}$* $\Longleftrightarrow$ существует полиномиальный алгоритм, который решает задачу P.

Замечание. Формальным аналогом понятия полиномиального алгоритма является *полиномиальная программа для детерминированной машины Тьюринга (ДМТ)*.

Класс $\mathcal{P}$ формально определяется как *класс всех языков, распознаваемых полиномиальными ДМТ-программами*.

Недетерминированные алгоритмы

Недетерминированный алгоритм состоит из двух стадий.

- На *стадии угадывания* по заданной индивидуальной задаче I происходит просто «угадывание» некоторой структуры S – *подсказки*.
- Затем I и S вместе подаются на вход *стадии проверки*, которая представляет собой обычный детерминированный алгоритм и либо заканчивается ответом «да», либо ответом «нет», либо продолжается бесконечно.

Комментарий. Подсказка S – это может быть предполагаемое решение задачи I или некоторая конструкция, позволяющая легко получить решение.

Класс $\mathcal{NP}$

Недетерминированный алгоритм *«решает»* задачу распознавания P, если $\forall I \in Z_P$ выполнено условие:

$I \in Y_P \Longleftrightarrow$ существует такая подсказка S, угадывание которой для входа I приводит к тому, что стадия проверки, начиная работу на входе (I, S), заканчивается ответом «да».

Недетерминированный алгоритм называется *полиномиальным*, если его стадия проверки – полиномиальный алгоритм.

Класс $\mathcal{NP}$ – это класс всех массовых задач распознавания, разрешимых недетерминированными алгоритмами за полиномиальное время.

Полиномиальная сводимость и NP-полные задачи

Пусть P, Q – две задачи распознавания и пусть A – полиномиальный алгоритм, который для любого входа $I \in Z_P$ строит некоторый вход $I' \in Z_Q$. Если при этом

$$I \in Y_P \Longleftrightarrow I' \in Y_Q,$$

то говорят, что задача P *полиномиально сводится* к задаче Q. *Обозначение:* $P <_K Q$.

Задача распознавания Q называется *NP-полной*, если $Q \in \mathcal{NP}$ и $P <_K Q \ \ \forall P \in \mathcal{NP}$.

Оптимизационные задачи

Оптимизационная задача P может быть охарактеризована тройкой объектов $< Z_P, S_P, f_P >$, где

Z_P – множество всех *индивидуальных задач (примеров)*,

S_P – функция, которая каждому примеру I ставит в соответствие множество $S_P(I)$ *допустимых решений* примера I,

f_P – *целевая функция*, определенная для всех пар (I, S), где $I \in Z_P$, $S \in S_P(I)$, а $f_P(I, S)$ – натуральное число.

Требуется $\forall I \in Z_P$ найти некоторое оптимальное решение $S^* \in S_P(I)$ и оптимальное значение $f^* = f_P(I, S^*)$.

Распознавательная версия оптимизационной задачи

Каждой оптимизационной задаче P *соответствует задача распознавания* P_b, в которой дополнительно задана граница $b \in \mathbb{N}$ и спрашивается, существует ли допустимое решение S примера I со значением

$f_P(I, S) \leqslant b$, если P – *задача минимизации*,

$f_P(I, S) \geqslant b$, если P – *задача максимизации*.

- Любая оптимизационная задача P *не проще* своей распознавательной версии P_b.

Класс $\mathcal{NPO}$

Оптимизационная задача $P =< Z_P, S_P, f_P >$ принадлежит *классу* $\mathcal{NPO}$, если

1) множество ее примеров Z_P распознаваемо за полиномиальное время;

2) существует полином g такой, что $\forall I \in Z_P$ и $\forall S \in S_P(I)$

длина кода $|S| \leqslant g(|I|)$ ($|I|$ – размер I).

Кроме того, $\forall I \in Z_P$ и $\forall S$ таких, что $|S| \leqslant g(|I|)$, за полиномиальное время разрешим вопрос $S \in S_P(I)$ или нет;

3) целевая функция f_P вычислима за полиномиальное время.

Класс $\mathcal{PO}$

Оптимизационная задача $P =< Z_P, S_P, f_P >$ принадлежит *классу* $\mathcal{PO}$, если

1) $P \in \mathcal{NPO}$;

2) существует полиномиальный алгоритм, который $\forall I \in Z_P$ находит оптимальное решение $S^* \in S_P(I)$ вместе с оптимальным значением $f^* = f_P(I, S^*)$.

Утверждение. 1) $\forall P \in \mathcal{PO}$ ее распознавательная версия принадлежит классу $\mathcal{P}$;

2) $\forall P \in \mathcal{NPO}$ ее распознавательная версия принадлежит классу $\mathcal{NP}$.

"$\mathcal{P}$ vs $\mathcal{NP}$" и "$\mathcal{PO}$ vs $\mathcal{NPO}$"

Так же, как для задач распознавания из класса $\mathcal{NP} \setminus \mathcal{P}$, ни для одной оптимизационной задачи из класса $\mathcal{NPO} \setminus \mathcal{PO}$ *не найдено* полиномиального алгоритма ее решения, но и *не доказано*, что такого алгоритма не существует.

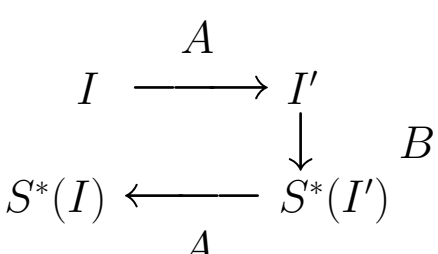
Вопросы $\mathcal{P} = \mathcal{NP}$? и $\mathcal{PO} = \mathcal{NPO}$? тесно связаны.

Можно показать, что эти вопросы *эквивалентны* в том смысле, что пололжительный ответ на один из них повлек бы положительный ответ на другой вопрос.

Сводимость по Тьюрингу

Пусть $P \in \mathcal{NPO}$ или $P \in \mathcal{NP}$, а $Q \in \mathcal{NPO}$.

Задача P *сводится по Тьюрингу* к задаче Q $(P <_T Q)$, если существует алгоритм A, который решает P с помощью (предполагаемой) процедуры B решения задачи Q, такой, что если B – полиномиальный алгоритм решения задачи Q, то A – полиномиальный алгоритм решения задачи P.



NP-трудные задачи

Оптимизационная задача Q называется *NP-трудной*, если $\forall P \in \mathcal{NP} \;\; P <_T Q$.

- Если $P \in \mathcal{NPO}$, а P_b – ее распознавательная версия, то $P_b <_T P$.

- Пусть $P \in \mathcal{NPO}$. Если ее распознавательная версия P_b NP-полна, то P является NP-трудной.

Теорема. Если $\mathcal{P} \neq \mathcal{NP}$, то ни одна из NP-трудных оптимизационных задач не может быть решена за полиномиальное время (а значит, $\mathcal{PO} \neq \mathcal{NPO}$).

Примеры NP-трудных задач

Дан граф $G = (V, E)$.

Множество $K \subseteq V$ называется *кликой*, если $\forall u, v \in K \;\; uv \in E$.

$H \subseteq V$ называется *независимым*, если $\forall u, v \in H \;\; uv \notin E$.

$B \subseteq V$ называется *вершинным покрытием*, если $\forall uv \in E$
$u \in B$ или $v \in B$.

КЛ. Найти в G клику максимальной мощности.

НМ. Найти в G независимое множество вершин максимальной мощности.

ВП. Найти в G вершинное покрытие минимальной мощности.

Эквивалентность трех задач

Лемма. $G = (V, E)$ – произвольный граф.
1) $K \subseteq V$ – клика в $G \iff K$ – независимое множество в дополнительном графе $\overline{G} = (V, \overline{E})$;
2) $H \subseteq V$ – независимое множество в $G \iff B = V \setminus H$ – вершинное покрытие в G.

Теорема. 1) ***КЛ*** $<_T$ ***НМ*** и наоборот;
2) ***НМ*** $<_T$ ***ВП*** и наоборот.

Следствие. Оптимизационные задачи ***КЛ***, ***НМ***, ***ВП*** эквивалентны с точки зрения поиска оптимальных решений.

Приближенные алгоритмы

Приближенным алгоритмом для оптимизационной задачи $P =< Z_P, S_P, f_P >$ называется любой алгоритм A, который $\forall I \in Z_P$ возвращает некоторое допустимое решение $S_A \in S_P(I)$ (если такое решение существует) вместе со значением целевой функции $f_P(I, S_A)$.

Далее нас будут интересовать только *полиномиальные приближенные алгоритмы*.

При этом хотелось бы заранее знать, как сильно полученное приближенное решение отличается от оптимального.

Другими словами нужны *априорные гарантированные оценки точности приближенных алгоритмов*.

$r(n)$-приближенные алгоритмы

Пусть $P \in \mathcal{NPO}$, A – приближенный алгоритм, который для любого $I \in Z_P$ находит допустимое решение $S_A \in S_P(I)$.

Обозн. $A(I) = f(I, S_A)$, $OPT(I) = f^*(I)$.

A называется *$r(n)$-приближенным алгоритмом* для P, где $r(n) : \mathbb{N} \to (1, \infty)$ – некоторая функция, если $\forall I \in Z_P$

$$\max\left(\frac{A(I)}{OPT(I)}, \frac{OPT(I)}{A(I)}\right) \leqslant r(n) \quad (n = |I|).$$

Если для задачи P существует $r(n)$-приближенный алгоритм, то говорят, что P *аппроксимируема с оценкой* $r(n)$.

$r(n)$ в этом случае называется *гарантированной оценкой точности* алгорима A.

Историческая справка

Понятие приближенного алгоритма с гарантированной оценкой точности было введено в 70-е годы XX века.

Среди первых работ в этом направлении можно отметить работы Garey-Graham-Ullman' 1972, Sahni' 1973, Johnson' 1974, Нигматуллин' 1976.

В те же годы Гимади, Глебов и Перепелица предложили наиболее общий подход к исследованию алгоритмов с оценками, включающий практически все основные понятия современной теории сложности приближенных вычислений.

Алгоритмы с оценками

Пусть P – *задача минимизации*, $\boldsymbol{Pr}$ – вероятностная мера, определенная на P. Для $I \in Z_P$ обозначим

$OPT(I)$ – оптимальное значение целевой функции,

$A(I)$ – значение, найденное алгоритмом A.

Алгоритм A *имеет оценки* (ε, δ) относительно $\boldsymbol{Pr}$, если $\forall I \in Z_P$

$$\boldsymbol{Pr}\left\{A(I) \leqslant (1+\varepsilon)OPT(I)\right\} \geqslant 1-\delta.$$

ε – *оценка относительной погрешности* алгоритма,

δ – *оценка вероятности несрабатывания* алгоритма A.

Для задачи P_n размера n говорят об оценках $(\varepsilon_n, \delta_n)$.

Алгоритмы с оценками

- Алгоритм с оценками $(0, 0)$ – *точный алгоритм*.
- Алгоритм с оценками $(\varepsilon, 0)$ является $(1+\varepsilon)$-*приближенным (имеет гарантированную оценку точности* $(1+\varepsilon)$*)*: $\forall\ I \in Z_P$

$$A(I) \leqslant (1+\varepsilon)OPT(I).$$

- Алгоритм A называется *асимптотически точным* для P, если существуют такие последовательности $(\varepsilon_n, \delta_n)$, что $\forall n$ алгоритм A имеет оценки $(\varepsilon_n, \delta_n)$ для задач P_n размера n, причем $\varepsilon_n \to 0,\ \delta_n \to 0$ при $n \to \infty$.
- Если при этом все $\varepsilon_n = 0$, то такой алгоритм *точен для почти всех задач* из P.

Трудные задачи

- Если $\mathcal{P} \neq \mathcal{NP}$, то для любой NP-трудной оптимизационной задачи не существует полиномиального алгоритма ее *точного решения*.

- Оказалось, что для некоторых NP-трудных задач даже задача поиска их $(1+a)$-*приближенных решений* остается NP-трудной для некоторой константы $a > 0$.

- Более того, для некоторых NP-трудных задач даже задача поиска их $r(n)$-*приближенных решений* остается NP-трудной, где $r(n) = c \log n$ или $r(n) = n^{\delta}$.

Классы аппроксимации

Оптимизационная задача $P \in \mathcal{NPO}$ *принадлежит классу* APX, если для нее существует $(1+a)$-приближенный алгоритм, где $a > 0$ – некоторая константа.

В классе APX выделяют *подкласс* $PTAS$ всех $\mathcal{NPO}$-задач, для которых существует полиномиальная приближенная схема.

Алгоритм A называется *полиномиальной приближенной схемой* для задачи $P \in \mathcal{NPO}$, если $\forall I \in Z_P$ и $\forall \varepsilon > 0$ алгоритм A является $(1+\varepsilon)$-приближенным алгоритмом

(т.е. A находит решение, сколь угодно близкое к оптимальному).

Классы аппроксимации

Оптимизационная задача P *принадлежит классу* $LogAPX$, если для нее существует $c \log n$-приближенный алгоритм, где $c > 0$ – некоторая константа.

Оптимизационная задача P *принадлежит классу* $PolyAPX$, если для нее существует n^{δ}-приближенный алгоритм, где $\delta > 0$ – некоторая константа.

Оптимизационная задача P *принадлежит классу* $ExpAPX$, если для нее существует 2^{n^k}-приближенный алгоритм, где $k > 0$ – некоторая константа.

Иерархия задач класса $\mathcal{NPO}$

$$\mathcal{PO} \subseteq PTAS \subseteq APX \subseteq LogAPX \subseteq PolyAPX \subseteq ExpAPX \subseteq \mathcal{NPO}$$

Теорема. Если $\mathcal{P} \neq \mathcal{NP}$, то все включения – строгие.

- **ВП** $\in APX$
- **КЛ**, **НМ** $\in PolyAPX$
- **КВ** $\in ExpAPX$
- Евклидова задача **КВ** $\in PTAS$
- Задача о покрытии множества **ПМ** $\in LogAPX$

Задача о покрытии множества (*ПМ*)

Постановка задачи. Дано конечное множество U, $|U| = n$, и семейство $S_1, ..., S_m$ его подмножеств, такое, что $\bigcup_{i=1}^{m} S_i = U$. Требуется найти подмножество $I \subseteq \{1, ..., m\}$ минимальной мощности, такое, что $\bigcup_{i \in I} S_i = U$.

- Задача ***ПМ*** NP-трудна, так как она является обобщением NP-трудной задачи ***ВП***.
- ***ПМ*** $\in LogAPX$, так как для нее существует $(\ln n + 1)$-приближенный алгоритм.

Классы неаппроксимируемости

Оптимизационная задача P *принадлежит классу I*, если задача поиска ее $(1+a)$-приближенного решения NP-трудна для некоторой константы $a > 0$
(т.е. для P не существует $(1+a)$-приближенного алгоритма).

Оптимизационная задача P *принадлежит классу II*, если задача поиска ее $c \log n$-приближенного решения NP-трудна для некоторой константы $c > 0$
(т.е. для P не существует $c \log n$-приближенного алгоритма).

Класс III определяется аналогично: $P \in III$, если задача поиска ее n^{δ}-приближенного решения NP-трудна для некоторой константы $\delta > 0$
(т.е. для P не существует n^{δ}-приближенного алгоритма).

Иерархия NP-трудных задач

$$\mathcal{NPO} \supseteq I \supseteq II \supseteq III$$

Теорема. Если $\mathcal{P} \neq \mathcal{NP}$, то все включения – строгие.

- **ВП** $\in I \quad \Longrightarrow$ **ВП** $\in APX \setminus PTAS$
- **ПМ** $\in II \quad \Longrightarrow$ **ПМ** $\in LogAPX \setminus APX$
- **КЛ**, **НМ** $\in III \quad \Longrightarrow$ **КЛ**, **НМ** $\in PolyAPX \setminus LogAPX$

Сводная иерархия задач класса $\mathcal{NPO}$