Н. М. Новикова

ОСНОВЫ ОПТИМИЗАЦИИ
(курс лекций)

МОСКВА 1998

Ответственный редактор
академик РАН П. С. Краснощеков

В сжатой форме дается изложение основ теории сложности, линейного программирования (ЛП) — с описанием полиномиальных алгоритмов, целочисленного ЛП, математического программирования (необходимые условия экстремума при ограничениях-неравенствах, локальные методы безусловной оптимизации, метод штрафов, идеи глобальной оптимизации), схем методов динамического программирования и ветвей и границ.

Работа написана на базе семестрового курса лекций, читаемого автором студентам 4-го курса программистского потока факультета ВМиК МГУ, с учетом дополнений и замечаний, указанных студентами. Автор благодарит всех студентов, содействовавших изданию этого курса и предложивших исправления, способствующие его улучшению, в том числе, Ласкавого Сергея, Санникова Андрея и Свахина Николая. Замеченные опечатки и неточности просьба сообщать автору по адресу nnovik@ccas.ru

Работа частично поддержана грантом РФФИ No.96-01-00786.

Рецензенты: С. К. Завриев,
А. В. Лотов

Часть 1. ВВЕДЕНИЕ В ТЕОРИЮ СЛОЖНОСТИ

Литература:

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

2. Пападимитриу Х., Стайглиц К. Комбинаторная оптимизация. М.: Мир, 1985.

§1. Понятие о сложности решения задач

Основные определения: индивидуальная и массовая задачи, кодировка, алгоритм решения массовой задачи, временна́я сложность алгоритма. Классы **P** *и* **NP** *(формальные определения, примеры).*

1. На вопрос, для чего надо иметь представление о сложности решаемых задач, наиболее наглядный ответ дан во введении к книге [1]. В этой книге также приводится более 500 задач (из самых разных областей, включая теорию графов и сетей, теорию расписаний, теорию автоматов и языков, оптимизацию программ, базы данных, игры и головоломки и т.п.), для которых в настоящее время нет оснований надеяться построить эффективные алгоритмы их решения. Что это значит формально, будет рассказано в данном разделе (§§1–4), а соответствующие практические выводы каждый человек, так или иначе связанный с разработкой алгоритмов и программ, делает для себя сам. Кроме того, теория сложности — новая, модная, интенсивно развивающаяся область математики и кибернетики, ее терминология широко распространена в современной научной литературе и требует определенного с ней знакомства.

Появление вычислительной техники привело к тому, что все реже приходится решать отдельную конкретную задачу, а все больше писать программы, рассчитанные на целый класс задач, получающихся одна из другой заменой ряда исходных данных. Поэтому имеет смысл говорить о сложности не для одной *индивидуальной* задачи **I**, а для *массовой задачи, или проблемы* **П**, соответствующей множеству индивидуальных задач.

Формально массовая задача **П** определяется

1^0 общим списком всех параметров задачи (свободных параметров, значения которых не заданы),

2^0 формулировкой свойств, которым должен удовлетворять ответ (решение задачи).

Индивидуальная задача $\mathbf{I} \in \mathbf{\Pi}$ получается из $\mathbf{\Pi}$, если всем параметрам присвоить конкретные значения.

Для примера рассмотрим задачу коммивояжера: найти минимальный маршрут обхода группы объектов (условно говоря, городов) с возвратом в начальную точку. Для $\mathbf{\Pi}$ коммивояжера введем

1^0 входные параметры: число городов m или множество городов $C = \{c_1, \ldots, c_m\}$ и набор расстояний между городами

$$\{d(c_i, c_j) > 0 : \quad c_i, c_j \in C, \quad i \neq j\};$$

2^0 требования к решению: $[c_{\pi(1)}, \ldots, c_{\pi(m)}]$ реализует

$$\min_{\pi} \left[\sum_{i=1}^{m-1} d(c_{\pi(i)}, c_{\pi(i+1)}) + d(c_{\pi(m)}, c_{\pi(1)}) \right],$$

где минимум берется по всем возможным перестановкам π на множестве индексов городов. Конкретизируем параметры 1^0, чтобы получить индивидуальную задачу $\mathbf{I}$ коммивояжера: $m = 4$, $d(c_1, c_2) = 10$, $d(c_1, c_3) = 5$, $d(c_1, c_4) = 9$, $d(c_2, c_4) = 9$, $d(c_3, c_4) = 3$, $d(c_3, c_2) = 6$, $d(c_i, c_j) = d(c_j, c_i)$. Тогда в задаче $\mathbf{I}$ оптимальным оказывается маршрут $[c_1, c_2, c_4, c_3]$, реализующий путь минимальной длины 27.

Кроме первичных понятий массовой и индивидуальной задачи ($\mathbf{\Pi}$ и $\mathbf{I}$) мы будем использовать термин алгоритм и обозначение $\mathbf{A}$ для пошаговой процедуры (решения задачи), в частности машины Тьюринга или программы для ЭВМ. Будем говорить, что *алгоритм* $\mathbf{A}$ *решает массовую задачу* $\mathbf{\Pi}$, если для любой индивидуальной задачи $\mathbf{I} \in \mathbf{\Pi}$ алгоритм $\mathbf{A}$ применим к $\mathbf{I}$ (т.е. останавливается за конечное число шагов) и $\forall \mathbf{I} \in \mathbf{\Pi}$ алгоритм $\mathbf{A}$ дает решение задачи $\mathbf{I}$. Например, для $\mathbf{\Pi}$ коммивояжера существует алгоритм, который решает ее на основе полного перебора всех маршрутов (перестановок π).

Большинство дискретных и комбинаторных задач допускает решение с помощью некоторого процесса перебора вариантов, однако число возможных вариантов растет экспоненциально в зависимости от размеров задачи (так, в задаче коммивояжера $m!$ маршрутов).

Поэтому переборные алгоритмы решения массовых задач считаются неэффективными (могут решать лишь небольшие индивидуальные задачи). В отличие от них эффективными называются *полиномиальные алгоритмы* решения массовой задачи, т.е. такие, которые решают произвольную $\mathbf{I} \in \mathbf{\Pi}$ за время, ограниченное полиномом от "размера" $\mathbf{I}$. Несмотря на определенную условность этого разделения с точки зрения практического счета, оно объясняется прежде всего тем, что центральным для дискретной оптимизации является вопрос, можно ли построить алгоритм решения массовой задачи (т.е. любой индивидуальной), не перебирающий всех или почти всех вариантов ее решения. Если для массовой задачи $\mathbf{\Pi}$ существует полиномиальный алгоритм, ее решающий, значит, ее можно решить не путем перебора — эффективно. Указанные задачи $\mathbf{\Pi}$ называются полиномиальными. Перейдем к их формальному определению.

2. Формализация проводится для *задач распознавания свойств*. Это — массовые задачи, предполагающие ответ "да" или "нет" в качестве решения. Таким образом, в п.2^0 определения $\mathbf{\Pi}$ распознавания свойств стоит некоторый альтернативный вопрос и решением каждой индивидуальной задачи $\mathbf{I} \in \mathbf{\Pi}$ является правильное распознавание, принадлежит ли она к задачам с ответом "да". Последнее подмножество множества индивидуальных задач будем обозначать $\mathbf{Y}$. Теперь введем обозначение $\mathbf{D}$ для множества всех возможных значений параметров, заданных в п.1^0 определения $\mathbf{\Pi}$. Очевидно, что набор $[\mathbf{D(\Pi)},\mathbf{Y(\Pi)}]$ полностью характеризует соответствующую массовую задачу $\mathbf{\Pi}$ распознавания свойств. Несмотря на специфичность постановки, класс задач распознавания свойств является достаточно широким: по крайней мере, для любой задачи дискретной оптимизации можно указать аналогичную $\mathbf{\Pi}$ распознавания свойств. В частности, для $\mathbf{\Pi}$ коммивояжера, если ввести в п.1^0 еще один параметр B — длину маршрута, то вопрос в п.2^0 "существует ли маршрут длины, не превышающей B?" дает ее переформулировку как задачи распознавания свойств. Полученная $\mathbf{\Pi}$ коммивояжера имеет в литературе обозначение $\mathbf{KM}$ (или ЗК [2]), для нее

$$\mathbf{D(KM)} = \{C, \quad \{d(c_i, c_j) \in \mathbf{Z}_+ |\ c_i, c_j \in C,\ i < j\}, \quad B \in \mathbf{Z}_+\}.$$

Здесь и далее $\mathbf{Z}_+$ — множество натуральных чисел, $\mathbf{Z}$ — целых.

Для формализации "размера" индивидуальной задачи свяжем с

каждой проблемой $\mathbf{\Pi}$ определенную *схему кодирования (кодировку)*. Введем конечное множество — *алфавит* $\Sigma = \{\sigma_i\}$, например $\Sigma = \{0,\ 1\}$, а также множество Σ^* *слов над алфавитом* Σ — произвольных конечных последовательностей, составленных из символов алфавита, возможно повторяющихся, $\sigma = \sigma_{i_1}\sigma_{i_2}\ldots\sigma_{i_n},\quad \sigma_{i_j} \in \Sigma\ \ \forall i_j$; например, пустое множество $\emptyset$ или 011000. Число n называется *длиной слова* σ и обозначается знаком модуля, $n = |\sigma|$. *Кодировкой задачи* $\mathbf{\Pi}$ назовем такое отображение e: $\mathbf{\Pi} \to \Sigma^*$, ставящее в соответствие любой индивидуальной задаче $\mathbf{I} \in \mathbf{\Pi}$ ее код $e(\mathbf{I}) = \sigma \in \Sigma^*$ (слово из алфавита Σ^*), что

1^* возможно однозначное декодирование: $\forall \mathbf{I}_1 \neq \mathbf{I}_2\ \ e(\mathbf{I}_1) \neq e(\mathbf{I}_2)$;

2^* $e,\ e^{-1}$ полиномиально вычислимы: существует алгоритм, реализующий $e,\ e^{-1}$, и полином $p(\cdot)$, для которого $\forall \mathbf{I} \in \mathbf{\Pi}$ время определения $e(\mathbf{I})$ и $e^{-1}(e(\mathbf{I}))$ не превосходит $p(|e(\mathbf{I})|)$;

3^* кодировка неизбыточна: для любой другой кодировки e', удовлетворяющей условиям 1^*,2^*, найдется полином $p'(\cdot)$ такой, что $\forall \mathbf{I} \in \mathbf{\Pi}\ \ |e(\mathbf{I})| < p'(|e'(\mathbf{I})|)$. Например, для записи целых чисел неизбыточной является любая k-ичная система счисления с $k > 1$, кодировка чисел тем же количеством палочек (случай $k = 1$) избыточна.

УПРАЖНЕНИЕ 1. Предложить неизбыточную кодировку и оценить по порядку длину входа задачи коммивояжера, сравнить полученную оценку с указанной в [1] на с. 35:

$$m + \lceil \log_2 B \rceil + \max\{\lceil \log_2 d(c_i, c_j)\rceil |\ c_i, c_j \in C\}.$$

Здесь и далее знаком $\lceil\cdot\rceil$ обозначается ближайшее целое сверху к числу в скобках, а $\lfloor\cdot\rfloor$ — целая часть числа.

Начиная с этого момента, в §§ 1–3 мы будем, как правило, рассматривать $\mathbf{\Pi}$ распознавания свойств, оговаривая другие случаи особо.

После того как для массовой задачи $\mathbf{\Pi}$ введена кодировка, с любой индивидуальной задачей $\mathbf{I} \in \mathbf{\Pi}$ будет связано некоторое слово σ в алфавите Σ этой кодировки. Слова, которые соответствуют индивидуальным задачам распознавания свойств, имеющим ответ “да”, условимся считать “правильными” и множество правильных слов в Σ^* назовем *языком*. Формально, язык $L(\mathbf{\Pi},e) \doteq e(\mathbf{Y}(\mathbf{\Pi}))\doteq$

$$\doteq \{\sigma \in \Sigma^* |\ \Sigma \text{ — алфавит } e,\quad \sigma = e(\mathbf{I}),\quad \mathbf{I} \in \mathbf{Y}(\mathbf{\Pi})\}.$$

С алгоритмом $\mathbf{A}$ решения задачи $\mathbf{\Pi}$ распознавания свойств будем ассоциировать машину Тьюринга (программу для детермини-

рованной машины Тьюринга) с входным алфавитом Σ и конечными состояниями q_Y ("да") и q_N ("нет") и аналогично назовем языком алгоритма $\mathbf{A}$ множество слов, *принимаемых* $\mathbf{A}$ (с которыми на входе $\mathbf{A}$ останавливается в состоянии q_Y — "да"),

$$L(\mathbf{A}) \doteq \{\sigma \in \Sigma^* |\ \Sigma \text{ — алфавит } \mathbf{A}, \text{ и } \mathbf{A}(\sigma) = q_Y\}.$$

Определение 1. Алгоритм $\mathbf{A}$ *решает* массовую задачу $\mathbf{\Pi}$ с кодировкой e, если $L(\mathbf{A}) = L(\mathbf{\Pi},e)$ и $\forall\sigma \in \Sigma^*$ $\mathbf{A}$ останавливается. Обозначим $t_{\mathbf{A}}(\sigma)$ время работы над словом $\sigma \in \Sigma^*$ (число шагов) алгоритма $\mathbf{A}$ до остановки. *Временно́й сложностью* алгоритма $\mathbf{A}$ решения массовой задачи $\mathbf{\Pi}$ назовем функцию $T_{\mathbf{A}}(\cdot)$, определяемую как

$$T_{\mathbf{A}}(n) = \max_{\sigma\in\Sigma^*:\ |\sigma|<n} t_{\mathbf{A}}(\sigma) \quad \forall n \in \mathbf{Z}_+.$$

Таким образом, при оценке временно́й сложности алгоритмов мы рассчитываем на "худшую" из возможных индивидуальных задач (данного размера), поскольку заранее не известно, с какой конкретной задачей придется работать.

УПРАЖНЕНИЕ 2. Дать алгоритм распознавания простоты числа, оценить временну́ю сложность алгоритма.

Определение 2. *Класс полиномиально разрешимых задач* $\mathbf{P} \doteq \{L(\mathbf{\Pi},e)|\ \exists\mathbf{A}$, решающий $\mathbf{\Pi}$ с кодировкой e, $\exists p(\cdot)$ — полином: $T_{\mathbf{A}}(n) < p(n) \quad \forall n \in \mathbf{Z}_+\}$.

Если для задачи $\mathbf{\Pi}$ существует такая кодировка e, что $L(\mathbf{\Pi},e) \in \mathbf{P}$, то будем называть задачу $\mathbf{\Pi}$ *полиномиально разрешимой* или просто *полиномиальной* и пользоваться обозначением $\mathbf{\Pi}\in\mathbf{P}$, отождествляя массовую задачу и язык. (С учетом условия неизбыточности кода указанная процедура корректна, ибо для полиномиальной $\mathbf{\Pi}$ получаем $L(\mathbf{\Pi},e) \in \mathbf{P}$ $\forall e$.)

Примером полиномиальной задачи является распознавание четности целого числа. (С еще одной полиномиальной задачей мы встретимся в разд.2.) Для задачи распознавания простоты числа (**ПЧ**) вопрос о ее полиномиальности пока открыт. Для ряда других задач удается доказать их неполиномиальность. Так, известны

1) алгоритмически неразрешимые задачи, когда не существует алгоритма, решающего любую индивидуальную задачу, т.е. $\forall\mathbf{A}$ $\exists\mathbf{I} \in \mathbf{\Pi}$: $\mathbf{A}$ не применим к $\mathbf{I}$, в частности, $t_A(e(\mathbf{I})) = \infty$; например, 10-я проблема Гильберта: по данному многочлену g с целыми

коэффициентами выяснить, имеет ли уравнение $g = 0$ целочисленное решение (неразрешимость доказал Ю. М. Матиясевич в 1970 г.);

2) задачи (не являющиеся задачами распознавания свойств), для которых длина записи решения превосходит любой наперед заданный полином от длины входа, например в задаче коммивояжера, если требуется найти все маршруты (их экспоненциальное число);

3) в остальных случаях формально имеем $\forall \mathbf{A}$, решающего $\mathbf{\Pi}$ с кодировкой e, $\forall p(\cdot)\ \exists \mathbf{I} \in \mathbf{\Pi}$: $t_{\mathbf{A}}(e(\mathbf{I})) > p(|e(\mathbf{I})|)$. Здесь и далее $p(\cdot)$, возможно с индексами, служит для обозначения полиномов.

В настоящее время для любой массовой задачи $\mathbf{\Pi}$, для которой доказано последнее условие, получен и более сильный результат: отсутствие полиномиального алгоритма, использующего произвольное (пусть бесконечное) число параллельных процессоров. Вопрос, существуют ли неполиномиальные задачи $\mathbf{\Pi}$ распознавания свойств, которые оказываются полиномиально разрешимыми при возможности распараллеливания вычислений, является *основной* методологической *проблемой* теории сложности (обусловившей ее формирование как самостоятельной научной дисциплины). Ответ, по-видимому, должен быть положительным, и уже указан большой класс массовых задач в качестве кандидатов (см. класс $\mathbf{NPC}$ в §2), но доказать или опровергнуть эту гипотезу в данный момент представляется нереальным. Для ее формализации вводится объемлющий $\mathbf{P}$ класс *недетерминированно полиномиальных задач* — $\mathbf{NP}$.

3. Определим *недетерминированную машину Тьюринга* (НДМТ) $\hat{\mathbf{A}}$ как набор обычных — детерминированных — машин Тьюринга (ДМТ) $\mathbf{A}(S)$ с алфавитом Σ, где S пробегает все множество слов из Σ^*:

$$\hat{\mathbf{A}} \doteq \{\mathbf{A}(S)\}_{S \in \Sigma^*}.$$

НДМТ $\hat{\mathbf{A}}$ останавливается, когда останавливается первая из ДМТ $\mathbf{A}(S)$, принимающая входное слово. Соответствующим конечным состоянием будет q_Y. *Язык НДМТ* — множество слов, принимаемых хотя бы одной ДМТ $\mathbf{A}(S)$ из $\hat{\mathbf{A}}$:

$$\hat{L}(\hat{\mathbf{A}}) \doteq \{\sigma \in \Sigma^* |\ \exists S \in \Sigma^* :\ \ \sigma \in L(\mathbf{A}(S))\}.$$

Слова S в определении НДМТ можно проинтерпретировать как подсказки к решению (догадки), тогда ДМТ $\mathbf{A}(S)$ проверяет для

входного слова σ подсказку S и в случае правильности останавливается в состоянии q_Y. НДМТ $\hat{\mathbf{A}}$ проверяет для входного слова σ все возможные подсказки, и если хоть одна правильная догадка существует, то НДМТ останавливается с ответом "да". (В силу бесконечности числа догадок, в состоянии q_N НДМТ остановиться не может.)

Определение 3. НДМТ $\hat{\mathbf{A}}$ *решает* массовую задачу $\mathbf{\Pi}$ с кодировкой e, если $L(\mathbf{\Pi},e) = \hat{L}(\hat{\mathbf{A}})$, т.е. языки НДМТ и задачи совпадают: $\forall\sigma \in L(\mathbf{\Pi},e)\ \exists S \in \Sigma^*$: ДМТ $\mathbf{A}(S)$ останавливается в состоянии q_Y, и $\forall\sigma \in \Sigma^* \setminus L(\mathbf{\Pi},e)$, $\forall S \in \Sigma^*$ ДМТ $\mathbf{A}(S)$ не принимает σ (не останавливается или останавливается в состоянии q_N).

Определим $\forall\sigma \in \hat{L}(\hat{\mathbf{A}})$ *время работы НДМТ* $\hat{\mathbf{A}}$ над словом σ как минимальное из времен работы над входом σ ДМТ $\mathbf{A}(S)$, принимающих σ, с учетом времени прочтения слова S (т.е. его длины):

$$\hat{t}_{\hat{\mathbf{A}}}(\sigma) \doteq \min_{\{S|\ \sigma \in L_{\mathbf{A}(S)}\}} \{|S|\ + t_{\mathbf{A}(S)}(\sigma)\}.$$

Временно́й сложностью НДМТ $\hat{\mathbf{A}}$ решения массовой задачи $\mathbf{\Pi}$ назовем функцию $\hat{T}_{\hat{\mathbf{A}}}(\cdot)$: $\quad \forall n \in \mathbf{Z}_+$

$$\hat{T}_{\hat{\mathbf{A}}}(n) = \max_{\sigma \in \hat{L}(\hat{\mathbf{A}}):\ |\sigma|<n} \hat{t}_{\hat{\mathbf{A}}}(\sigma) \doteq \max_{\sigma \in \hat{L}(\hat{\mathbf{A}}):\ |\sigma|<n}\ \min_{\{S|\sigma \in L_{\mathbf{A}(S)}\}} \{|S| + t_{\mathbf{A}(S)}(\sigma)\}.$$

Подчеркнем разницу с определением временно́й сложности ДМТ: для НДМТ рассматриваются лишь слова из языка (соответствующие индивидуальным задачам с ответом "да").

Определение 4. *Класс недетерминированно полиномиальных задач* $\mathbf{NP} \doteq \{L(\mathbf{\Pi},e)|\ \exists\hat{\mathbf{A}}$ — НДМТ, решающая $\mathbf{\Pi}$ с кодировкой e, $\exists p(\cdot)$ — полином: $\hat{T}_{\hat{\mathbf{A}}}(n) < p(n) \quad \forall n \in \mathbf{Z}_+\}$. Если для задачи $\mathbf{\Pi}$ существует такая кодировка e, что $L(\mathbf{\Pi},e) \in \mathbf{NP}$, то будем называть задачу $\mathbf{\Pi}$ *недетерминированно полиномиальной* и пользоваться обозначением $\mathbf{\Pi}\in\mathbf{NP}$ (как и для класса $\mathbf{P}$, корректным).

Примером недетерминированно полиномиальной задачи является $\mathbf{KM}$, ибо в качестве догадки можно использовать маршрут и проверка его допустимости полиномиальна.

Отметим, что полиномиальность проверки гарантируется только для индивидуальных задач с ответом "да" (и возможно, лишь при

единственной подсказке), а для $\mathbf{I} \in \mathbf{D}(\mathbf{\Pi}) \setminus \mathbf{Y}(\mathbf{\Pi})$ НДМТ просто не остановится. В этом — существенное отличие классов $\mathbf{P}$ и $\mathbf{NP}$. Непосредственно из определений следует

Утверждение 1. $\mathbf{P} \subseteq \mathbf{NP}$.

Вопрос о наличии строгого включения и является формализацией основной проблемы теории сложности.

§2. NP-полные (универсальные) задачи

Теорема об экспоненциальной оценке временно́й сложности для задач из класса $\mathbf{NP}$. *Класс* $\mathbf{co\text{-}NP}$. *Задачи, имеющие хорошую характеризацию. Определение полиномиальной сводимости. Класс* $\mathbf{NPC}$. *Теорема Кука (без доказательства). Критерий* $\mathbf{NP}$*-полноты. Доказательство* $\mathbf{NP}$*-полноты задачи* **БЛН** *(булевы линейные неравенства).*

1. Рассмотрим подробнее класс $\mathbf{NP}$.

Теорема 1. Для любой недетерминированно полиномиальной задачи существует ДМТ, решающая ее с экспоненциальной временной сложностью, т.е. $\forall \mathbf{\Pi} \in \mathbf{NP}$ $\exists p(\cdot)$ — полином — и ДМТ $\mathbf{A}$:

$$\mathbf{A} \text{ решает } \mathbf{\Pi} \text{ и } T_{\mathbf{A}}(n) < 2^{p(n)} \quad \forall n \in \mathbf{Z}_+.$$

Доказательство. Так как $\mathbf{\Pi} \in \mathbf{NP}$, то для любого слова σ из языка задачи $\mathbf{\Pi}$ существует правильная догадка S полиномиальной длины: $|S| < p_1(|\sigma|)$, $p_1(\cdot)$ — полином, и существует ДМТ $\mathbf{A}(S)$: $t_{\mathbf{A}(S)}(\sigma) < p_2(|\sigma|)$, $p_2(\cdot)$ — полином. Построим ДМТ $\mathbf{A}$, которая работает над любым входным словом $\sigma \in \Sigma^*$ (с любой индивидуальной задачей $\mathbf{I} \in \mathbf{\Pi}$) следующим образом: рассматриваются все слова S из Σ^* длины меньше $p_1(|\sigma|)$ и делается не более $p_2(|\sigma|)$ шагов с каждой ДМТ $\mathbf{A}(S)$. Если очередная ДМТ останавливается в состоянии q_Y (т.е. соответствующая догадка оказалась правильной), считаем слово σ принятым и работу ДМТ $\mathbf{A}$ законченной; если ни одна из ДМТ $\mathbf{A}(S)$ не остановилась за отведенное время или остановилась в состоянии q_N, то заканчиваем работу ДМТ $\mathbf{A}$ и приписываем ей конечное состояние q_N. В последнем случае ДМТ $\mathbf{A}$ делает наибольшее число шагов, и это число меньше $p_2(|\sigma|) \cdot |\Sigma|^{p_1(|\sigma|)}$ (второй сомножитель равен числу проверяемых догадок, $|\Sigma|$ — число символов в алфавите Σ). Отсюда уже нетрудно получить утверждение теоремы.

Для того чтобы лучше почувствовать различие классов **P** и **NP**, введем понятие *дополнительной к* **П** массовой задачи $\overline{\mathbf{П}}$, получающейся из **П** распознавания свойств заменой альтернативного вопроса, определяющего ответ в задаче (см. п.2^0 определения **П** в §1) его отрицанием, например вопросом в $\overline{\mathbf{КМ}}$ "правда ли, что не существует маршрута длины, не превосходящей B?". Формально $\mathbf{D}(\overline{\mathbf{П}}) = \mathbf{D}(\mathbf{П})$, $\mathbf{Y}(\overline{\mathbf{П}}) = \mathbf{D}(\mathbf{П}) \setminus \mathbf{Y}(\mathbf{П})$.

Определим классы дополнительных задач $\mathbf{co\text{-}P} \doteq \{\overline{\mathbf{П}} |\ \mathbf{П} \in \mathbf{P}\}$ и $\mathbf{co\text{-}NP} \doteq \{\overline{\mathbf{П}} |\ \mathbf{П} \in \mathbf{NP}\}$. Из определений очевидно, что, если ДМТ **A** решает **П**, то ДМТ $\overline{\mathbf{A}}$ решает $\overline{\mathbf{П}}$, где программа ДМТ $\overline{\mathbf{A}}$ получена из программы ДМТ **A** простой заменой конечных состояний q_Y и q_N друг на друга. Таким образом, справедливо

Утверждение 2. $\mathbf{co\text{-}P} = \mathbf{P}$.

Аналогичное утверждение для класса **NP** до сих пор не удается ни доказать ни опровергнуть: приведенное выше для ДМТ рассуждение нельзя обобщить на НДМТ, ибо для индивидуальных задач **I** с ответом "нет" (т.е. $\mathbf{I} \notin \mathbf{Y}(\mathbf{П})$, или $\mathbf{I} \in \mathbf{Y}(\overline{\mathbf{П}})$) НДМТ не останавливается за время, ограниченное полиномом от длины входа **I**. В частности, не известна НДМТ, решающая $\overline{\mathbf{КМ}}$ за полиномиальное время, так как для нее не придумано подсказки полиномиальной длины (естественный вариант — показать все маршруты — не полиномиален); включение $\overline{\mathbf{КМ}} \in \mathbf{NP}$ не доказано и не опровергнуто.

УПРАЖНЕНИЕ 3. Доказать, что задача распознавания простоты числа принадлежит классу co-NP, т.е. $\overline{\mathbf{ПЧ}} \in \mathbf{NP}$.

Определение 5. Массовая задача распознавания свойств называется *имеющей хорошую характеризацию*, если для нее выполнено $\mathbf{П} \in \mathbf{NP} \cap \mathbf{co\text{-}NP}$.

Из утверждения 2 следует, что $\mathbf{P} \subseteq \mathbf{NP} \cap \mathbf{co\text{-}NP}$. Современная гипотеза состоит в равенстве этих классов. Отсюда и термин "хорошая характеризация", так как для подобных задач есть основания надеяться на возможность построения полиномиальных алгоритмов (см. задачу **ЛН** — линейные неравенства — в разд.2). Однако для задачи **ПЧ**, обладающей хорошей характеризацией (для доказательства того, что $\mathbf{ПЧ} \in \mathbf{NP}$, см. [2, с. 414]), детерминированного полиномиального алгоритма пока не найдено, несмотря на ее непосредственную практическую значимость.

2. Задач распознавания свойств — большое разнообразие, и для теории представляет интерес не только возможность их классификации, но и способы определения класса сложности одних задач на основе известного класса сложности других. Поэтому вводится базовое для теории сложности понятие *полиномиальной сводимости*.

Определение 6. Будем говорить, что массовая задача распознавания свойств $\mathbf{\Pi}'$ с кодировкой e' *полиномиально сводится* к задаче $\mathbf{\Pi}$ с кодировкой e, если любая индивидуальная задача $\mathbf{I}' \in \mathbf{\Pi}'$ может быть сведена за полиномиальное от ее длины время к некоторой $\mathbf{I} \in \mathbf{\Pi}$ с сохранением ответа. Формально

существует функция сводимости $f:\ e'(\mathbf{D}(\mathbf{\Pi}')) \to e(\mathbf{D}(\mathbf{\Pi}))$, такая что $f(e'(\mathbf{Y}(\mathbf{\Pi}'))) = e(\mathbf{Y}(\mathbf{\Pi}))$, т.е. $\forall \sigma' \in e'(\mathbf{Y}(\mathbf{\Pi}'))\ f(\sigma') \in e(\mathbf{Y}(\mathbf{\Pi}))$ и $\forall \sigma'' \in e'(\mathbf{D}(\mathbf{\Pi}') \setminus \mathbf{Y}(\mathbf{\Pi}))\ \ f(\sigma'') \in e(\mathbf{D}(\mathbf{\Pi}) \setminus \mathbf{Y}(\mathbf{\Pi}))$,

и существует ДМТ $\mathbf{A_f}$, реализующая f за полиномиальное время, т.е. $\exists p_f(\cdot)$ — полином: $\forall \sigma \in e'(\mathbf{D}(\mathbf{\Pi}'))\ \ t_{\mathbf{A_f}}(\sigma) < p_f(|\sigma|)$.
В случае, когда соответствующие кодировки не избыточны, будем использовать термин полиномиальной сводимости по отношению к самим задачам (без указания кодировок) и применять обозначение

$$\mathbf{\Pi}' \propto \mathbf{\Pi}.$$

(Корректность упрощения вытекает из полиномиальной сводимости задачи к самой себе, но с другой неизбыточной кодировкой, и следующего очевидного утверждения — транзитивности отношения $\propto$.)

Утверждение 3. Если $\mathbf{\Pi}_1 \propto \mathbf{\Pi}_2$ и $\mathbf{\Pi}_2 \propto \mathbf{\Pi}_3$, то $\mathbf{\Pi}_1 \propto \mathbf{\Pi}_3$.

Существенным для теории сложности является

Утверждение 4. Если $\mathbf{\Pi}' \propto \mathbf{\Pi}$ и $\mathbf{\Pi} \in \mathbf{P}$, то и $\mathbf{\Pi}' \in \mathbf{P}$.

Доказательство. Обозначим $\mathbf{A}$ ДМТ, решающую $\mathbf{\Pi}$ с полиномиальной временной сложностью, и построим ДМТ $\mathbf{A}'$, решающую $\mathbf{\Pi}'$ с полиномиальной временной сложностью, как суперпозицию ДМТ $\mathbf{A}$ и $\mathbf{A_f}$: $\mathbf{A}' = \mathbf{A} \circ \mathbf{A_f}$, т.е. сначала к любому входному слову $\sigma' \in e'(\mathbf{D}(\mathbf{\Pi}'))$ применяется $\mathbf{A_f}$, а потом к получившемуся слову $\sigma = f(\sigma')$ (длиной не более $p_f(|\sigma'|)$) применяется $\mathbf{A}$. Временная сложность $\mathbf{A}'$ — $T_{\mathbf{A}'}(\cdot) \le T_{\mathbf{A_f}}(\cdot) + T_{\mathbf{A}}(p_f(\cdot))$ — полином.

Аналогично доказывается (при замене слова ДМТ на НДМТ)

Утверждение 5. Если $\mathbf{\Pi}' \propto \mathbf{\Pi}$ и $\mathbf{\Pi} \in \mathbf{NP}$, то и $\mathbf{\Pi}' \in \mathbf{NP}$.

Определение 7. Массовая задача $\mathbf{\Pi}$ называется **NP**-*полной или универсальной*, если $\mathbf{\Pi} \in \mathbf{NP}$ и $\forall \mathbf{\Pi}' \in \mathbf{NP} \quad \mathbf{\Pi}' \propto \mathbf{\Pi}$ (т.е. любая недетерминированно полиномиальная задача полиномиально сводится к $\mathbf{\Pi}$). Класс всех **NP**-полных задач (распознавания свойств) обозначается **NPC** (**NP**-complete).

Непустоту класса **NPC** доказал С. А. Кук в 1971 г. Им была рассмотрена задача о выполнимости (**ВЫП**): выяснить выполнимость конъюнктивной нормальной формы (КНФ) — конъюнкции конечного числа дизъюнктивных функций, т.е. дизъюнкций булевых переменных z_i или их отрицаний $\overline{z_i}$. А именно, в задаче **ВЫП** требуется распознать для КНФ на входе, существует ли выполняющий набор z^0 (для которого значение КНФ равно 1).

Теорема 2 (S. A. Cook). **ВЫП** $\in$ **NPC**.

Доказательство полиномиальной сводимости к **ВЫП** любой недетерминированно полиномиальной задачи основано на формальной записи условия принадлежности слова σ языку из класса **NP** (того, что σ принимается некоторой НДМТ, а значит, и какой-то ДМТ) в виде набора дизъюнктивных функций от специально вводимых булевых переменных, связанных с состояниями ДМТ в различные моменты времени, и является недостаточно простым для вводного курса (см. [1,2]). Поэтому мы лишь убедимся в том, что **ВЫП** $\in$ **NP**. Действительно, входное слово (параметры, определяющие индивидуальную задачу выполнимости) содержит число дизъюнктивных функций в КНФ и указание для каждой из них, какие переменные входят с отрицанием, а какие не входят вообще. Длину такого слова можно ограничить снизу суммой длин дизъюнктивных функций, понимая под длиной функции число ее переменных (или число знаков дизъюнкции + 1). Если теперь в качестве подсказки для определяемой входным словом КНФ взять z^0 — выполняющий ее набор, то вычисление на нем значения КНФ (проверка выполнимости) потребует такого же по порядку числа шагов.

Из определения **NP**-полноты непосредственно следует

Утверждение 6. Если $\mathbf{P} \cap \mathbf{NPC} \neq \emptyset$, то $\mathbf{P} = \mathbf{NP}$. А если $\mathbf{NPC} \cap (\mathbf{NP} \setminus \mathbf{P}) \neq \emptyset$, то $\mathbf{NPC} \subseteq \mathbf{NP} \setminus \mathbf{P}$.

Таким образом, если бы удалось найти полиномиальный алгоритм решения хоть одной **NP**-полной задачи, то были бы построены

полиномиальные алгоритмы решения всех **NP**-полных задач и всех задач из класса **NP**, а если для какой-либо **NP**-полной задачи доказать отсутствие полиномиального алгоритма ее решения, то это не только дает строгое включение $\mathbf{P} \subset \mathbf{NP}$ (т.е. ответ к основной проблеме теории сложности), но и влечет за собой доказательство невозможности построения полиномиального алгоритма решения любой задачи из класса **NPC**. Поскольку ни того, ни другого пока не сделано, считается, что задачи из **NPC** отвечают житейскому представлению о трудной задаче и вряд ли допускают эффективное решение. Поэтому, если встречается задача, для которой на практике не удается придумать неперебoрный алгоритм, то имеет смысл попытаться доказать ее **NP**-полноту, чтобы оправдать применение к ней тех или иных переборных схем.

3. После того как была установлена непустота класса **NPC** (теоремой Кука), появилась возможность доказательства **NP**-полноты массовой задачи $\mathbf{\Pi}$ путем полиномиального сведения к $\mathbf{\Pi}$ одной из известных **NP**-полных задач (соответствующий список см. в [1]). Действительно, из утверждения 3 следует

Теорема 3 (*критерий* **NP**-*полноты*). Массовая задача $\mathbf{\Pi}$ распознавания свойств **NP**-полна тогда и только тогда, когда она принадлежит классу **NP** и к ней полиномиально сводится какая-либо **NP**-полная задача:

$$\{\mathbf{\Pi} \in \mathbf{NPC}\} \iff \{\mathbf{\Pi} \in \mathbf{NP} \text{ и } \exists \mathbf{\Pi}' \in \mathbf{NPC}: \ \mathbf{\Pi}' \propto \mathbf{\Pi}\}.$$

Пользуясь теоремой 3, можно показать **NP**-полноту задачи о существовании целочисленного решения системы линейных неравенств с целыми коэффициентами (**ЦЛН**).

Утверждение 7. $\mathbf{ЦЛН} \in \mathbf{NPC}$.

Доказательство. 1) $\mathbf{ЦЛН} \in \mathbf{NP}$, так как подсказкой может служить решение системы, а его проверка сводится к умножению на заданные коэффициенты и сложению, что не превосходит полинома от длины записи всех коэффициентов (доказательство полиномиальности длины записи решения см. в [2, с. 330]).

2) $\mathbf{ВЫП} \propto \mathbf{ЦЛН}$. Общий вид системы линейных неравенств

$$a_{j1}z_1 + a_{j2}z_2 + \ldots + a_{jn}z_n \le b_j, \quad j = 1, \ldots, m.$$

Нетрудно представить в подобной форме условие истинности дизъюнктивной функции. Для этого заменим в каждой j-й функции знаки

дизъюнкции знаками суммы, а отрицания переменных z_i — на $(1-z_i)$ и напишем для получившейся линейной функции условие ≥ 1, добавив ограничения $z_i \geq 0$ и $z_i \leq 1$ на все переменные. Целочисленное решение $z^0 = \{z_i^0\}$ системы всех построенных неравенств является выполняющим набором для исходной КНФ (так как истинность КНФ эквивалентна истинности всех образующих ее дизъюнктивных функций). Таким способом решение любой индивидуальной задачи о выполнимости сводится к решению некоторой индивидуальной задачи $\mathbf{I} \in \mathbf{ЦЛН}$. Полиномиальность сведения очевидна.

Заметим, что фактически в п.2 данного доказательства доказан более сильный результат о сведении **ВЫП** к подзадаче **ЦЛН** — задаче о существовании булева решения системы линейных неравенств с целыми коэффициентами (**БЛН**). Доказательство принадлежности **БЛН** классу недетерминированно полиномиальных задач повторяет п.1 данного доказательства без ссылки на [2] (так как полиномиальность длины булева решения очевидна), тем самым получено и

Утверждение 8. $\mathbf{БЛН} \in \mathbf{NPC}$.

§3. Классы сложности. Сильная NP-полнота и псевдополиномиальность

Доказательство **NP***-полноты задачи о 3-выполнимости. Взаимоотношение классов* **P, NP** *и* **NPC, NP** *и* **co-NP**. **NP***-трудные задачи. Класс* **PSPACE**. *Псевдополиномиальные алгоритмы. Пример для задачи о рюкзаке. Сильная* **NP***-полнота. Теорема о связи сильной* **NP***-полноты задачи с существованием псевдополиномиального алгоритма ее решения.*

1. Кроме задачи о выполнимости, **NP**-полнота всех остальных известных задач из класса **NPC** (в том числе и **КМ**) была доказана на основе теоремы 3 с помощью полиномиального сведения. Общие рецепты доказательства полиномиальной сводимости (см. в [1]) легко использовать лишь в простейших случаях. Чтобы научиться их применять, надо разобрать большое число примеров (в частности, имеющиеся в [1,2]), на что у нас в рамках данной работы нет возможности. Однако, еще один пример будет далее приведен с целью показать, что не только любая подзадача сводится к соответствующей задаче (автоматически), но возможно и обратное сведение.

Рассмотрим частный случай задачи о выполнимости, когда в КНФ могут входить лишь дизъюнктивные функции трех переменных (**3-ВЫП**). Поскольку **D(3-ВЫП)⊂D(ВЫП)**, то по определению **3-ВЫП∝ВЫП**. Так что **3-ВЫП∈NP** (по утверждению 5). Но ее **NP**-полнота требует специального доказательства, ибо частные массовые задачи содержат меньше индивидуальных задач и могут оказаться проще; например, аналогичная задача **2-ВЫП** полиномиальна. Для получения результата **3-ВЫП** $\in$ **NPC** докажем, что **NP**-полная задача о выполнимости сводится к своей подзадаче (частному случаю) **3-ВЫП**.

Утверждение 9. **ВЫП** $\propto$ **3-ВЫП**.

Доказательство. Покажем, что произвольную дизъюнктивную функцию $f^j(z^j)$ k переменных можно представить в виде конъюнкции дизъюнктивных функций от трех переменных (за счет введения дополнительных переменных u^j). Обозначим через y_i переменную z_i^j или $\overline{z}_i^j$ в зависимости от того, как i-я компонента z^j входит в рассматриваемую дизъюнктивную функцию; тогда последнюю можно записать как $y_1 \vee y_2 \vee \ldots \vee y_k$ и при $k > 3$ заменить на КНФ:

$$(y_1 \vee y_2 \vee u_1^j)\&(y_3 \vee \overline{u_1^j} \vee u_2^j)\&(y_4 \vee \overline{u_2^j} \vee u_3^j)\& \ldots$$
$$\ldots \&(y_{k-2} \vee \overline{u_{k-4}^j} \vee u_{k-3}^j)\&(y_{k-1} \vee y_k \vee \overline{u_{k-3}^j}).$$

Отметим, что данная замена не является эквивалентной. Действительно, если исходная дизъюнктивная функция равнялась нулю, то построенная КНФ равна нулю при всех значениях u, но если исходная дизъюнктивная функция равнялась 1, то найдется такое значение u, чтобы КНФ равнялась 1. Этого, однако, достаточно для сохранения ответа на вопрос о существовании выполняющего набора.

УПРАЖНЕНИЕ 4. Завершить доказательство NP-полноты задачи 3-ВЫП (рассмотреть случаи $k < 3$).

2. Универсальность задач из класса **NPC** (**NP**-полных задач) состоит в том, что основные нерешенные вопросы для класса **NP** (недетерминированно полиномиальных задач) достаточно разрешить хотя бы для одной **NP**-полной задачи, чтобы получить ответ для всего класса **NP**. Кроме утверждения 6 здесь также важно

Утверждение 10. Если для некоторой **NP**-полной задачи **П** дополнительная к ней $\overline{\mathbf{П}}$ принадлежит классу **NP**, то **NP = co-NP**.

Доказательство. Так как $\mathbf{\Pi} \in \mathbf{NPC}$, то $\forall \mathbf{\Pi}' \in \mathbf{NP}\ \ \mathbf{\Pi}' \propto \mathbf{\Pi}$, отсюда и $\overline{\mathbf{\Pi}}' \propto \overline{\mathbf{\Pi}}$ (полиномиальное сведение осуществляется той же функцией — см. определение 6). Но $\overline{\mathbf{\Pi}} \in \mathbf{NP}$, значит, $\overline{\mathbf{\Pi}}' \in \mathbf{NP}$ по утверждению 5. С учетом произвольности $\mathbf{\Pi}' \in \mathbf{NP}$ получили, что $\mathbf{co\text{-}NP} \subseteq \mathbf{NP}$. Обратное включение доказывается на основании очевидного равенства $\mathbf{\Pi} = \overline{\overline{\mathbf{\Pi}}}$.

Напомним, что гипотеза $\mathbf{NP} = \mathbf{co\text{-}NP}$ в настоящее время кажется нереальной (реальная гипотеза $\mathbf{P} = \mathbf{NP} \cap \mathbf{co\text{-}NP}$), и вряд ли для какой-либо **NP**-полной задачи удастся доказать принадлежность классу **co-NP**. Поэтому для конкретной недерминированно полиномиальной массовой задачи $\mathbf{\Pi} \in \mathbf{NP}$, если ее решение представляет интерес, имеет смысл попытаться доказать включение $\overline{\mathbf{\Pi}} \in \mathbf{NP}$ (т.е. ее хорошую характеризацию) и затем построить полиномиальный алгоритм решения, либо, когда указанное включение не доказывается, надо попробовать полиномиально свести к $\mathbf{\Pi}$ одну из известных **NP**-полных задач (т.е. показать **NP**-полноту $\mathbf{\Pi}$) и в случае успеха подыскивать переборную схему решения (учитывая ограничения на размерность практически решаемых индивидуальных задач).

Доказательство универсальности $\mathbf{\Pi}$, т.е. включения $\mathbf{\Pi} \in \mathbf{NPC}$, удается не всегда, и в теории сложности был введен объемлющий **NPC** класс **NP**-*трудных* задач, содержащий

1) $\mathbf{\Pi}$ распознавания свойств, для которых доказано, что $\mathbf{\Pi}' \propto \mathbf{\Pi}$ для $\mathbf{\Pi}' \in \mathbf{NPC}$, но не показано, что $\mathbf{\Pi} \in \mathbf{NP}$ (в частности, это задачи $\mathbf{\Pi} \in \mathbf{co\text{-}NPC}$);

2) $\mathbf{\Pi}$ оптимизации, для которых соответствующие $\mathbf{\Pi}$ распознавания свойств **NP**-полны (например, задача коммивояжера из п.1 §1);

3) и остальные массовые задачи (не обязательно распознавания свойств), к которым *сводятся по Тьюрингу* какие-либо **NP**-полные задачи $\mathbf{\Pi}' \in \mathbf{NPC}$, где сводимость по Тьюрингу — $\mathbf{\Pi}' \propto_T \mathbf{\Pi}$ — означает, что из существования полиномиального алгоритма решения $\mathbf{\Pi}$ следует существование полиномиального алгоритма и для $\mathbf{\Pi}'$.

Задачи $\mathbf{\Pi}$ (не обязательно распознавания свойств), для которых $\exists \mathbf{\Pi}' \in \mathbf{NPC}$: $\mathbf{\Pi}' \propto_T \mathbf{\Pi}$ и $\exists \mathbf{\Pi}'' \in \mathbf{NP}$: $\mathbf{\Pi} \propto_T \mathbf{\Pi}''$, называются задачами из класса **NP**-*эквивалентных*.

Все рассмотренные нами классы задач **П** — *классы сложности* включаются в общий класс **PSPACE** массовых задач (не обязательно распознавания свойств), для решения которых существуют алгоритмы, требующие не более чем полиномиальной памяти. Вопрос о равенстве **P** и **PSPACE** является открытым. Рабочая гипотеза состоит в наличии строгого включения $\mathbf{P} \subset \mathbf{PSPACE}$, при этом **NP**-полные, **NP**-трудные и **NP**-эквивалентные задачи должны включаться в $\mathbf{PSPACE} \setminus \mathbf{P}$. (Соответствующую диаграмму взаимосвязи классов сложности см. в [2, с. 412].)

Заметим, что теоретически рассмотренную схему построения классов сложности можно применять и для других схем классификации; в частности, вводят также класс **PSPACE**-полных задач (к которым полиномиально сводится любая задача из класса **PSPACE**). Кроме полиномиальной сводимости можно аналогично говорить о логарифмической сводимости, о задачах, требующих логарифмической памяти и т.п. В настоящее время наиболее интенсивно изучаемым здесь оказывается класс **NC** (Nick Class, автор N. Pippenger) задач, решаемых за время, ограниченное полиномом от логарифма длины входа, и требующих полиномиальной памяти (логарифмическое время при возможности полиномиального числа процессоров). К сожалению, изложение полученных для **NC** результатов выходит за рамки введения в теорию сложности.

3. Ранее уже отмечалось, что с практической точки зрения разница между полиномиальным алгоритмом (для полиномов высоких степеней) и экспоненциальным весьма условна, а все дело в возможности или невозможности избежать полного перебора. Вопрос, все ли **NP**-полные и **NP**-трудные задачи трудны для практического счета, мы обсудим ниже в этом параграфе.

Рассмотрим самую простую (по формулировке) **NP**-трудную задачу — задачу о рюкзаке (**ЗР**), заключающуюся в том, чтобы из имеющегося набора полезных вещей собрать рюкзак ограниченного объема с наибольшей пользой. Формально требуется найти

$$\max_{z:\ z_j \in \{0,1\}\ \forall j=1,\dots,n} \{\sum_{j=1}^{n} c_j z_j \mid \sum_{j=1}^{n} w_j z_j \le K\},$$

где $c_j \in \mathbf{Z}_+$ — полезность (ценность), $w_j \in \mathbf{Z}_+$ — объем (вес) j-й вещи, а переменная z_j определяет класть или не класть ее в рюкзак; максимально возможный объем (вес) рюкзака задается параметром $K \in \mathbf{Z}_+$. Соответствующая задача распознавания свойств — существует ли булево решение системы двух линейных неравенств

$$\sum_{j=1}^{n} c_j z_j \geq B \quad \text{и} \quad \sum_{j=1}^{n} w_j z_j \leq K$$

с натуральными коэффициентами — **NP**-полна (доказательство не следует из утверждения 8, так как рассматривается частный случай **БЛН**, поэтому см. [1, с. 87 или 2, с. 386]). Для решения **ЗР** известен следующий метод (*динамического программирования*).

Произвольная индивидуальная задача **I**$\in$**ЗР** погружается в семейство задач поиска

$$F_i(E) \doteq \max_{z:\ z_j \in \{0,1\}\ \forall j=i,\ldots,n} \{\sum_{j=i}^{n} c_j z_j \mid \sum_{j=i}^{n} w_j z_j \leq K - E\},$$

$F_1(0)$ — значение **ЗР**. И решаются все задачи данного семейства по рекуррентным формулам, где i убывает с n до 1. А именно, положим $F_i(E) \doteq 0 \quad \forall E \geq K, \quad \forall i$. Имеем $\forall E = \overline{0, K-1}$:

$$F_n(E) = \begin{cases} 0, & E > K - w_n, \\ c_n & \text{иначе}, \end{cases}$$

и $\forall i = n-1, \ldots, 2$: $F_i(E) = \max\{F_{i+1}(E),\ c_i + F_{i+1}(E + w_i)\} \doteq$

$$\doteq \max_{z_i \in \{0,1\}} \{c_i z_i + F_{i+1}(E + w_i z_i)\}; \quad F_1(0) = \max\{F_2(0),\ c_1 + F_2(w_1)\}.$$

Число итераций предложенного алгоритма равно nK и того же порядка будет его временна́я сложность. Отметим, что алгоритм не является полиномиальным, ибо для записи числа K требуется порядка $\log_2 K$ символов; он также оказывается переборным — перебирает все варианты заполненности рюкзака. Однако при не очень больших объемах рюкзака можно довольно быстро получить решение. Для обобщения указанного свойства дадим

Определение 8. Обозначим через $\mathrm{num}(\mathbf{I})$ максимальное по модулю целое число (или 0), фигурирующее при задании числовых параметров для индивидуальной задачи $\mathbf{I}$, и через $|\mathbf{I}| \doteq |e(\mathbf{I})|$ — длину записи $\mathbf{I}$. Алгоритм $\mathbf{A}$ решения массовой задачи $\mathbf{\Pi}$ (не обязательно распознавания свойств) называется *псевдополиномиальным*, если для некоторого полинома $p(\cdot,\cdot)$ выполнено $t_{\mathbf{A}}(e(\mathbf{I})) < p(|\mathbf{I}|,\mathrm{num}(\mathbf{I}))$ $\forall \mathbf{I} \in \mathbf{\Pi}$.

Примером псевдополиномиального алгоритма является алгоритм динамического программирования для решения **ЗР**. Для многих других задач (в частности, **КМ**) псевдополиномиальных алгоритмов не известно. Чтобы выделить класс таких задач, введем понятие *полиномиального сужения* массовой задачи $\mathbf{\Pi}$ как множества тех индивидуальных задач, числовые параметры которых не превосходят полинома от длины входа,

$$\mathbf{\Pi}_{p(\cdot)} \doteq \{\mathbf{I} \in \mathbf{\Pi} |\ \mathrm{num}(\mathbf{I}) < p(|\mathbf{I}|)\}.$$

Определение 9. Массовая задача $\mathbf{\Pi}$ распознавания свойств называется *сильно* **NP**-*полной*, если ее полиномиальное сужение **NP**-полно, т.е. $\exists p(\cdot)$ — полином: $\mathbf{\Pi}_{p(\cdot)} \in \mathbf{NPC}$.

Примерами сильно **NP**-полных задач являются **ВЫП** и **3-ВЫП** (как совпадающие со своими полиномиальными сужениями), **БЛН** (поскольку **ВЫП** была сведена к ее полиномиальному сужению, в котором модули правых частей не превышают n), **ЦЛН** (как обобщение **БЛН** в отличие от **ЗР**), а также **КМ** [1, с. 123-124].

Теорема 4. Если $\mathbf{P} \neq \mathbf{NP}$, то ни для какой сильно **NP**-полной задачи не существует псевдополиномиального алгоритма решения.

Доказательство проведем от противного. Пусть ДМТ $\mathbf{A}$ решает сильно **NP**-полную задачу $\mathbf{\Pi}$ и $\forall \mathbf{I} \in \mathbf{\Pi}$ $t_{\mathbf{A}}(e(\mathbf{I})) < p'(|\mathbf{I}|,\mathrm{num}(\mathbf{I}))$ для полинома $p'(\cdot,\cdot)$. Тогда $\forall \mathbf{I} \in \mathbf{\Pi}_{p(\cdot)}$ $t_{\mathbf{A}}(e(\mathbf{I})) < p'(|\mathbf{I}|, p(|\mathbf{I}|)) = p''(|\mathbf{I}|)$, т.е. $\mathbf{\Pi}_{p(\cdot)} \in \mathbf{P}$ — противоречие с $\mathbf{\Pi}_{p(\cdot)} \in \mathbf{NPC}$ или утверждением 6.

Сильно **NP**-полные задачи считаются наиболее трудными для счета среди всех задач класса **NP**. Далее мы покажем, что для подобных задач в оптимизационной постановке отсутствуют эффективные алгоритмы поиска даже приближенного решения. Рекомендуемым подходом к их решению является разбиение на подзадачи $\mathbf{\Pi}'$: $\mathbf{D}(\mathbf{\Pi}') \subseteq \mathbf{D}(\mathbf{\Pi})$, $\mathbf{Y}(\mathbf{\Pi}') = \mathbf{Y}(\mathbf{\Pi}) \cap \mathbf{D}(\mathbf{\Pi}')$, анализ сложности подзадач

и разработка схем перебора (см. в §§10,12) для $\mathbf{\Pi}' \in \mathbf{NPC}$. При этом для сильно **NP**-полных подзадач не удается использовать метод динамического программирования в качестве способа перебора (ибо, по-видимому, реализующие его алгоритмы псевдополиномиальны) и следует ориентироваться на схему метода ветвей и границ (§§10,11).

§4. Приближенное решение задач комбинаторной оптимизации

Определение задачи комбинаторной оптимизации и приближенного алгоритма ее решения. Утверждение о разнице между приближенным и точным оптимумом для задачи о рюкзаке. Определение ε-приближенного алгоритма и полностью полиномиальной приближенной схемы (ППТС). Связь между существованием ППТС и псевдополиномиальностью. Теорема об отсутствии ППТС для задач оптимизации, соответствующих сильно **NP***-полным задачам распознавания свойств. Пример задачи о коммивояжере.*

Важный класс массовых задач образуют *задачи дискретной (комбинаторной) оптимизации.* Для оптимизационной постановки задачи $\mathbf{\Pi}$ решением каждой индивидуальной задачи $\mathbf{I}\in\mathbf{\Pi}$ является произвольная реализация оптимума

$$Opt_{\mathbf{\Pi}}(\mathbf{I}) \doteq \max_{z \in S_{\mathbf{\Pi}}(\mathbf{I})} f_{\mathbf{\Pi}}(\mathbf{I}, z),$$

т.е. такая точка $z^*(\mathbf{I}) \in S_{\mathbf{\Pi}}(\mathbf{I})$, для которой $f_{\mathbf{\Pi}}(\mathbf{I}, z^*(\mathbf{I})) = Opt_{\mathbf{\Pi}}(\mathbf{I})$. Здесь $S_{\mathbf{\Pi}}(\mathbf{I}) \subseteq \mathbf{Z}^{n(I)}$ — область допустимых значений дискретной (целочисленной) переменной z, $f_{\mathbf{\Pi}}(\mathbf{I}, \cdot) : S_{\mathbf{\Pi}}(\mathbf{I}) \to \mathbf{Z}$ — целевая функция индивидуальной задачи $\mathbf{I}$ оптимизации, знак max в постановке задачи может быть заменен на min.

Будем обозначать σ_S и σ_f те компоненты входного слова $\sigma = e(\mathbf{I})$, определяющего параметры индивидуальной задачи $\mathbf{I} \in \mathbf{\Pi}$, которые задают допустимую область (ограничения задачи) и функцию цели соответственно. Например, для **ЗР** имеем $f_{\mathbf{ЗР}}(\sigma, z) = \langle c, z\rangle$, $S_{\mathbf{ЗР}}(\sigma) = \{z = (z_1, \ldots, z_n) | z_j \in \{0,1\}\ \forall j = \overline{1,n}$ и $\langle w, z\rangle \le K\}$, $\sigma_S = (n, w, K)$ и $\sigma_f = c$. Здесь и далее знак $\langle \cdot, \cdot \rangle$ обозначает скалярное произведение векторов.

Определение 10. Алгоритм $\mathbf{A}$ называется *приближенным алгоритмом решения массовой задачи* $\mathbf{\Pi}$ *оптимизации*, если $\forall \mathbf{I} \in \mathbf{\Pi}$ он находит некоторую точку из допустимой области $z_{\mathbf{A}}(\mathbf{I}) \in S_{\mathbf{\Pi}}(\mathbf{I})$, принимаемую за приближенное решение. Значение $f_{\mathbf{\Pi}}(\mathbf{I}, z_{\mathbf{A}}(\mathbf{I}))$ называется приближенным значением оптимума и обозначается $A(\mathbf{I})$.

Говорить об абсолютной погрешности приближенного алгоритма решения массовой задачи оптимизации (на классе всевозможных индивидуальных задач) не имеет большого смысла, как показывает

Утверждение 11. Если $\mathbf{P} \neq \mathbf{NP}$, то ни для какой константы $C > 0$ не существует полиномиального приближенного алгоритма $\mathbf{A}$ решения $\mathbf{ЗР}$ с оценкой $|Opt_{\mathbf{ЗР}}(\mathbf{I}) - A(\mathbf{I})| \leq C \quad \forall \mathbf{I} \in \mathbf{ЗР}$.

Доказательство проведем от противного. Пусть найдены такие C и $\mathbf{A}$. Построим алгоритм $\mathbf{A}'$ следующим образом: $\forall \mathbf{I} \in \mathbf{ЗР}$ домножим все коэффициенты c_j на $C+1$ — получим индивидуальную задачу $\mathbf{I}' \in \mathbf{ЗР}$, к которой применим алгоритм $\mathbf{A}$ и разделим полученный ответ на $C+1$, т.е. $A'(\mathbf{I}) = A(\mathbf{I}')/(C+1)$. Очевидно, $Opt_{\mathbf{ЗР}}(\mathbf{I}') = (C+1)Opt_{\mathbf{ЗР}}(\mathbf{I})$ и из полиномиальности алгоритма $\mathbf{A}$ вытекает полиномиальность $\mathbf{A}'$. При этом его точность равна $|Opt_{\mathbf{ЗР}}(\mathbf{I}) - A'(\mathbf{I})| = |Opt_{\mathbf{ЗР}}(\mathbf{I}') - A(\mathbf{I}')|/(C+1) \leq C/(C+1) < 1$, т.е. равна нулю (так как все значения целевой функции целые). Получили полиномиальный алгоритм точного решения $\mathbf{ЗР}$. Проверка $Opt_{\mathbf{ЗР}}(\mathbf{I}) \geq B$ полиномиальна, значит, построили и полиномиальный алгоритм решения $\mathbf{ЗР}$ в постановке распознавания свойств, что с учетом универсальности последней противоречит утверждению 6.

Определение 11. Приближенный алгоритм $\mathbf{A}$ решения массовой задачи $\mathbf{\Pi}$ оптимизации называется *ε-приближенным алгоритмом* решения $\mathbf{\Pi}$ для некоторого $\varepsilon > 0$, если

$$\forall \mathbf{I} \in \mathbf{\Pi} \qquad \frac{|Opt_{\mathbf{\Pi}}(\mathbf{I}) - A(\mathbf{I})|}{|Opt_{\mathbf{\Pi}}(\mathbf{I})|} < \varepsilon,$$

т.е. его относительная погрешность не превосходит ε.

Для ε-приближенных алгоритмов приведем следующий результат [2, с. 439], доказательство которого основано на методе динамического программирования и в данном курсе опускается.

Теорема 5. Пусть для задачи $\mathbf{\Pi}$ оптимизации

1) существует псевдополиномиальный алгоритм ее решения;

2) $\forall \mathbf{I} \in \mathbf{\Pi} \quad |Opt_{\mathbf{\Pi}}(\mathbf{I})| < p_1(|\mathbf{I}|, \mathrm{num}(\mathbf{I}))$ и $\mathrm{num}(\mathbf{I}) < p_2(|\mathbf{I}|, Opt_{\mathbf{\Pi}}(\mathbf{I}))$

для некоторых полиномов $p_1(\cdot,\cdot)$, $p_2(\cdot,\cdot)$;
3) $\forall\sigma = e(\mathbf{I})$, $\mathbf{I} \in \mathbf{\Pi}$: параметры σ_S, задающие ограничения, и σ_f, задающие целевую функцию, не пересекаются, и $\forall z \in S_{\mathbf{\Pi}}(\sigma)$ функция цели $f_{\mathbf{\Pi}}(\sigma, z)$ линейно зависит от параметров σ_f;

тогда $\exists p(\cdot,\cdot)$ — полином: $\forall\varepsilon > 0$ $\exists$ ε-приближенный алгоритм $\mathbf{A}_\varepsilon$ решения $\mathbf{\Pi}$ с временно́й сложностью $T_{\mathbf{A}_\varepsilon}(|\mathbf{I}|) < p(|\mathbf{I}|, 1/\varepsilon)$.

Теорема 5 справедлива, например, для **ЗР** (сравните результат с утверждением 11). Набор алгоритмов $\{\mathbf{A}_\varepsilon\}$, определенный в теореме 5, называется *полностью полиномиальной приближенной схемой* (ППТС) решения задачи $\mathbf{\Pi}$ оптимизации. Наличие ППТС — лучшее, чего можно ожидать при решении **NP**-трудных задач. К сожалению, в целом ряде случаев на это нельзя рассчитывать, так как имеется

Теорема 6. Если для $\mathbf{\Pi}$ оптимизации соответствующая ей $\mathbf{\Pi}$ распознавания свойств является сильно **NP**-полной и $\exists p'(\cdot)$ — полином: $|Opt_{\mathbf{\Pi}}(\mathbf{I})| < p'(\mathrm{num}(\mathbf{I}))$ $\forall\mathbf{I} \in \mathbf{\Pi}$, то при условии, что $\mathbf{P} \neq \mathbf{NP}$, для $\mathbf{\Pi}$ не существует ППТС.

Доказательство проведем от противного. Пусть ППТС существует. Построим алгоритм $\mathbf{A}'$ следующим образом. Для любой индивидуальной задачи $\mathbf{I}$ $\mathbf{A}'$ вызывает $\mathbf{A}_\varepsilon$ с $\varepsilon = 1/(p'(\mathrm{num}(\mathbf{I}))+1)$. Тогда по определению ε-приближенного алгоритма $\mathbf{A}_\varepsilon$ $|Opt_{\mathbf{\Pi}}(\mathbf{I}) - A'(\mathbf{I})| < |Opt_{\mathbf{\Pi}}(\mathbf{I})|/(p'(\mathrm{num}(\mathbf{I})) + 1) < p'(\mathrm{num}(\mathbf{I}))/(p'(\mathrm{num}(\mathbf{I})) + 1)$ по условию теоремы. Но в левой части полученного неравенства было целое число, которое оказывается равным нулю как неотрицательное, меньшее 1. Таким образом, алгоритм $\mathbf{A}'$ точен, причем $T_{\mathbf{A}'}(|\mathbf{I}|) = T_{\mathbf{A}_\varepsilon}(|\mathbf{I}|) < p(|\mathbf{I}|, p'(\mathrm{num}(\mathbf{I})) + 1)$ по определению ППТС. Следовательно, алгоритм $\mathbf{A}'$ псевдополиномиален, что противоречит теореме 4.

Утверждение 12. Если $\mathbf{P} \neq \mathbf{NP}$, то ни для какого $\varepsilon > 0$ не существует полиномиального ε-приближенного алгоритма решения оптимизационной постановки задачи коммивояжера.

Доказательство см. в [2, с. 440–441].

Для частного случая **КМ**, в котором функция $d(\cdot,\cdot)$ расстояния между городами удовлетворяет неравенству треугольника, известен 0.5-приближенный полиномиальный алгоритм Кристофидеса [2, с. 429–432] (решения **КМ** оптимизации).

Содержание