СИБИРСКИЙ ЖУРНАЛ ИССЛЕДОВАНИЯ ОПЕРАЦИЙ
Июль – сентябрь 1994. Том 1, № 3, 38–48

УДК 519.854

ПОЛИНОМИАЛЬНЫЙ В СРЕДНЕМ АЛГОРИТМ В ЦЕЛОЧИСЛЕННОМ ЛИНЕЙНОМ ПРОГРАММИРОВАНИИ*)

Н. Н. Кузюрин

Исследуется метод динамического программирования — один из общих методов, позволяющий в ряде случаев строить эффективные алгоритмы решения дискретных оптимизационных задач. Известно, что в худшем случае его сложность иногда является экспоненциальной. Доказано, что при некоторых условиях на распределение исходных данных задачи о многомерном рюкзаке метод динамического программирования дает алгоритм ее решения с полиномиальным средним временем. Показано также, что в худшем случае задачу линейного программирования с булевыми переменными (т. е. переменными, принимающими значения 0 и 1) невозможно аппроксимировать с мультипликативной точностью существенно ниже тривиальной, если класс $\mathcal{P}$ задач, решаемых за полиномиальное время на обычных машинах Тьюринга, не совпадает с классом $\mathcal{NP}$ задач, решаемых за полиномиальное время на недетерминированных машинах Тьюринга ($\mathcal{P} \neq \mathcal{NP}$).

Среди подходов к решению NP-трудных задач можно выделить два. Первый заключается в построении алгоритмов, позволяющих находить решения с гарантированными оценками точности [1], а второй — в отказе от анализа сложности алгоритмов по худшему случаю и переходе к анализу сложности в среднем [2, 3].

Первый подход в настоящее время активно исследуется, но полученные результаты в основном отрицательные [4–7]. Сформулировать их можно следующим образом: решения существенно лучшие, чем те, которые получаются с помощью простых естественных процедур (типа жадного (градиентного) алгоритма), не могут быть найдены в полиномиальное время, если $\mathcal{P} \neq \mathcal{NP}$. Например, для задачи о покрытии m-элементного множества жадный алгоритм дает решения, отличающиеся от минимального не более чем в $\ln m$ раз, а найти решения, отличающиеся от минимального не более чем в $(1/4)\log_2 m$ раз, невозможно за полиномиальное время, если $\mathcal{P} \neq \mathcal{NP}$ [7].

Второй подход по сравнению с оценками сложности по худшему случаю позволяет смягчить требования к эффективности алгоритмов

*) Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (код проекта 93-012-459).

и для ряда NP-полных задач построить «полиномиальные в среднем» алгоритмы (определение и примеры подобных результатов будут приведены ниже).

В настоящей работе указанные подходы применяются для решения следующей задачи целочисленного линейного программирования.

ЗАДАЧА О МНОГОМЕРНОМ РЮКЗАКЕ. Пусть элементы a_{ij} матрицы A размеров $m \times n$, компоненты b_j $(1 \leqslant j \leqslant m)$ вектора $\mathbf{b}$ длины m и компоненты c_i $(1 \leqslant i \leqslant n)$ вектора $\mathbf{c}$ длины n — произвольные действительные числа. Требуется найти максимум линейной функции $\langle \mathbf{c}, \mathbf{x} \rangle$ при следующих ограничениях:

$$A\mathbf{x} \leqslant \mathbf{b}, \qquad x_1, \dots, x_n \text{ — булевы переменные;} \tag{1}$$

здесь $\mathbf{x} \leqslant \mathbf{y}$ означает покомпонентное сравнение векторов.

Ниже (кроме п. 3) предполагаем, что a_{ij} и b_j, c_i $(1 \leqslant j \leqslant m$, $1 \leqslant i \leqslant n)$ суть целые неотрицательные числа. Линейная релаксация задачи (1) отличается от исходной задачи тем, что в ней отсутствует требование целочисленности переменных. Известно, что задача (1) в рассматриваемой постановке является NP-полной [1, 8]. В то же время задача о нахождении оптимума линейной релаксации принадлежит классу $\mathcal{P}$ [9].

Очевидно, вектор $(0, \dots, 0)$ длины n допустимый для задачи (1). Предположим, что векторы $\mathbf{c}$ и $\mathbf{b}$ фиксированы, а все элементы a_{ij} матрицы A суть независимые случайные величины, при этом ненулевые элементы a_{ij} удовлетворяют условию $\mathbf{P}\{a_{ij} = t\} = p$ при $t = 1, \dots, M$, а для нулевых a_{ij} имеем $\mathbf{P}\{a_{ij} = 0\} = 1 - pM > 0$, где $M = \max\limits_{1 \leqslant j \leqslant m} \{b_j\}$. Время работы алгоритма является случайной величиной, зависящей от исходных данных. *Средним временем работы алгоритма* называется математическое ожидание времени работы алгоритма. Алгоритм называется *полиномиальным в среднем*, если среднее время работы ограничено сверху некоторым полиномом от длины входа.

Отметим, что симплекс-метод решения задачи линейного программирования при анализе по худшему случаю является экспоненциальным [10], а при анализе в среднем — полиномиальным алгоритмом [8, 11, 12]. Для некоторых NP-полных задач известны полиномиальные в среднем алгоритмы [6, 13–15]. Однако имеются и трудные при анализе в среднем задачи [2, 3].

Отметим также, что понятие полиномиального в среднем алгоритма сильнее понятия алгоритма, который в полиномиальное время почти всегда (с вероятностью, стремящейся к единице) находит решение [16]. Многие алгоритмы, эффективные почти всегда, на самом деле являются экспоненциальными в среднем [15, 17].

Основной результат работы (теорема 1) сформулирован и доказан в п. 2. В п. 3 изложен результат о трудности нахождения хорошего приближенного решения задачи (1). В предположении $\mathcal{P} \neq \mathcal{NP}$ показано, что для любой константы δ $(0 < \delta < 1)$ задача (1) не может быть аппроксимирована в полиномиальное время с мультипликативной точностью $C(n - n^{\delta})$, где C — максимум модуля исходных данных.

1. Алгоритм. Метод динамического программирования для решения задачи (1) основан на рекуррентных соотношениях, позволяющих без полного перебора построить все допустимые решения. Опишем один из вариантов этого метода.

Для множества T векторов длины n и данного вектора $\mathbf{x}$ той же длины введем множество

$$T + \mathbf{x} = \{\mathbf{z} \mid \mathbf{z} = \mathbf{y} + \mathbf{x},\ \mathbf{y} \in T\}.$$

Через $\mathbf{a}^{(k)}$ обозначим k-й вектор-столбец матрицы A и через $D(k)$ — множество всех векторов $\mathbf{y}$ длины m таких, что

$$D(k) = \Big\{\mathbf{y} \mid \mathbf{y} \leqslant \mathbf{b}, \quad \mathbf{y} = \sum_{i=1}^{k} x_i \mathbf{a}^{(i)}, \quad x_i \text{ — булевы переменные}\Big\}.$$

Каждому вектору $\mathbf{y} \in D(k)$ сопоставим значение целевой функции $\langle \mathbf{c}, \mathbf{x} \rangle = \sum\limits_{i=1}^{k} c_i x_i$.

Рассмотрим следующую модификацию метода динамического программирования для решения задачи (1) (см. [8, 18]).

АЛГОРИТМ А

Шаг 1: $D(0) = \varnothing$.

Шаг 2: при $k = 1, 2, \ldots, n$ образовать множества $D(k+1) = D(k) \cup \{D(k) + \mathbf{a}^{(k+1)}\}$.

Шаг 3: выбрать в $D(n)$ вектор с максимальным значением целевой функции.

На шаге 2 в множество $D(k+1)$ включаются только допустимые векторы из множества $D(k) + \mathbf{a}^{(k+1)}$, удовлетворяющие ограничениям задачи (1). Кроме того, для реализации операции объединения на шаге 2 множество должно быть реализовано в виде некоторой динамической структуры данных, позволяющей за время $O(\log S)$ осуществлять проверку вхождения и размещение очередного элемента среди элементов множества мощности S (например, с помощью $2-3$ дерева [19]).

Нетрудно убедиться, что сложность описанного алгоритма есть величина порядка $O(n|D(n)|\log|D(n)|)$. В худшем случае она не превосходит $O(nm|D(n)|\log(M+1))$, поскольку $|D(n)| \leqslant (M+1)^m$. В п. 2

показано, что при некоторых условиях математическое ожидание размера $D(n)$ линейно и, следовательно, алгоритм является полиномиальным в среднем. Это вытекает из того факта, что среднее время работы алгоритма есть величина порядка $O(nm\mathrm{E}|D(n)|\log(M+1))$, равная числу операций сложения и сравнения векторов длины m. Поэтому для доказательства теоремы 1 достаточно оценить сверху математическое ожидание размера множества $D(n)$.

Нам удобно для доказательства теоремы 1 использовать следующую модификацию алгоритма А. Обозначим через $T(k)$ множество допустимых для задачи (1) булевых векторов с $n-k$ нулевыми последними компонентами и через $\mathbf{e}^{(k)}$ — вектор длины n такой, что его k-я компонента равна единице, а остальные — нулю.

АЛГОРИТМ В

Шаг 1: $T(0)=\varnothing$.

Шаг 2: при $k=1,2,\ldots,n$ образовать множества $T(k+1)=T(k)\cup\{T(k)+\mathbf{e}^{k+1}\}$.

Шаг 3: выбрать в множестве $T(n)$ вектор с максимальным значением целевой функции.

На шаге 2 в $T(k+1)$ так же, как в алгоритме А, включаются только допустимые векторы из $T(k)+\mathbf{e}^{k+1}$. Однако в отличие от алгоритма А здесь нет необходимости проверять вхождение элемента в множество, поскольку допустимые векторы заведомо различны. Поэтому в качестве структуры данных для $T(n)$ можно использовать обычный массив. При этом нетрудно убедиться, что сложность алгоритма В есть величина порядка $O(n|T(n)|)$. Отметим, что $|D(n)|\leqslant|T(n)|$. Среднее время работы алгоритма есть величина порядка $O(n\mathrm{E}|T(n)|)$, равная числу операций сложения и сравнения векторов длины m. Поэтому для доказательства теоремы 1 достаточно оценить сверху математическое ожидание мощности множества $T(n)$ (см. п. 2).

Для алгоритма А оценка получается несколько хуже, чем для алгоритма В, но при ее выводе не учитывается процесс «исключения дубликатов» на шаге 2, так что на практике алгоритм А может оказаться эффективнее алгоритма В. В обоих случаях полиномиальность в среднем является следствием линейности математического ожидания мощности множества $T(n)$.

2. Основной результат. В этом пункте мы формулируем и доказываем основной результат работы.

Теорема 1. *Пусть $m(pM)^2\geqslant c\ln n$ с некоторой достаточно большой константой c. Тогда имеется модификация метода динамического программирования для задачи* (1), *являющаяся полиномиальным в среднем алгоритмом.*

Доказательство. Для оценки среднего времени работы алгоритма B оценим сверху математическое ожидание числа допустимых решений задачи (1) (или мощность множества $T(n)$).

Вероятность $\mathbf{P}(k)$ того, что некоторый вектор с k единичными и $n-k$ нулевыми компонентами является допустимым решением задачи (1), можно оценить сверху следующим образом:

$$\mathbf{P}(k) \leqslant (p(k,M))^m, \tag{2}$$

где $p(k,M) = \mathbf{P}\{\sum_{i=1}^{k} a_{ij} \leqslant M\}$. Следовательно, для математического ожидания $\mathbf{E}|T(n)|$ мощности множества $T(n)$ справедлива оценка

$$\mathbf{E}|T(n)| \leqslant \sum_{k=0}^{n} \binom{n}{k} p(k,M)^m. \tag{3}$$

В свою очередь, имеем

$$p(k,M) = \sum_{R \leqslant M} \sum_{t=0}^{\min(k,R)} \binom{k}{t} p^t (1-pM)^{k-t} S(R,t), \tag{4}$$

где $S(R,t)$ — количество разбиений числа R на t упорядоченных натуральных слагаемых. Известно (см., например, [20, с. 261, 262]), что

$$S(R,t) = \binom{R-1}{t-1},$$

$$\sum_{R=0}^{M} \binom{R-1}{t-1} = \binom{M}{t}. \tag{5}$$

Пользуясь (4) и (5), получаем

$$p(k,M) = \sum_{t=0}^{k} \varphi(k,t), \tag{6}$$

где

$$\varphi(k,t) = \frac{1}{t!} \binom{k}{t} (pM)^t (1-pM)^{k-t}. \tag{7}$$

Будем различать два случая.

Случай 1: $kpM > 6$. Пусть $t_0 = \lfloor kpM/2 \rfloor$. Принимая во внимание (6), разобьем соответствующую сумму на две:

$$p(k,M) \leqslant S_1 + S_2,$$

где

$$S_1 = \sum_{0 \leqslant t \leqslant t_0} \binom{k}{t} (pM)^t (1-pM)^{k-t},$$
$$S_2 = \sum_{t_0 < t \leqslant k} \binom{k}{t} (pM)^t (1-pM)^{k-t} \frac{1}{t!}.$$

Чтобы оценить сверху величину S_1, используем неравенство для «хвостов» биномиального распределения (см. [16])

$$\sum_{t=0}^{(1-\delta)np} \binom{n}{t} p^t (1-p)^{n-t} \leqslant \exp(-\delta^2 np/2),$$

справедливое при любом δ, $0 < \delta < 1$. Применяя к S_1 эту оценку (при $\delta = 1/2$), получаем

$$S_1 \leqslant \exp(-kpM/8). \tag{8}$$

Используя неравенство $n! > e^{3n/4}$, справедливое при $n \geqslant 4$, выражение для S_2 оценим следующим образом:

$$S_2 \leqslant \frac{1}{t_0!} \leqslant \exp(-3kpM/8). \tag{9}$$

Подставляя (8) и (9) в (6), получаем

$$p(k,M) \leqslant \exp(-3kpM/8) + \exp(-kpM/8).$$

Поскольку функция $\varphi(x) = e^{-7x/24} + e^{-x/24}$ убывает при $x > 0$, а $\varphi(7) < 1$, при $x \geqslant 7$ имеем

$$e^{-x/12}(1 - e^{-7x/24} - e^{-x/24}) = e^{-x/12} - e^{-3x/8} - e^{-x/8} > 0.$$

Поэтому

$$p(k,M) \leqslant \exp(-kpM/12).$$

Следовательно, в силу (2) при $kpM > 6$ и $m > (12 \ln n)/p^2M^2$ получаем

$$\mathbf{P}(k) < (e^{-kpM/12})^m < n^{-k}. \tag{10}$$

Случай 2: $kpM \leqslant 6$. Из (6) и (7) следует, что

$$p(k,M) \leqslant (1-pM)^k + kpM(1-pM)^{k-1} + \frac{1}{2}\sum_{t=2}^{k}\binom{k}{t}(pM)^t(1-pM)^{k-t}$$

$$\leqslant \frac{1}{2}(1+(1-pM)^k+kpM(1-pM)^{k-1}) = \frac{1}{2}(1+(1-pM)^{k-1}(1+(k-1)pM))$$

$$\leqslant \frac{1}{2}(1+(1-pM)^{k-1}(1+pM)^{k-1}) = \frac{1}{2}(1+(1-(pM)^2)^{k-1})$$

$$\leqslant \frac{1}{2}(1+e^{-(pM)^2(k-1)}). \tag{11}$$

Рассмотрим равенство $e^{-y} = 1 - yf(y)$, где $f(y) = (1-e^{-y})/y$. С помощью производных убедимся в том, что на отрезке $0 < y \leqslant 6$ функция $f(y)$ монотонно убывает. Отсюда при $0 < y \leqslant 6$ следует неравенство

$$e^{-y} \leqslant 1 - (\frac{1}{6} - \frac{1}{6e^6})y,$$

подставляя которое в (11) при $y = (k-1)p^2M^2$, получим

$$p(k,M) \leqslant \frac{1}{2}\left[1+1-(k-1)p^2M^2\left(\frac{1}{6}-\frac{1}{6e^6}\right)\right] = 1-\left(\frac{1}{12}-\frac{1}{12e^6}\right)(k-1)p^2M^2.$$

Поэтому в силу (2) при $m \geqslant (13\ln n)/p^2M^2$ и $k \geqslant 2$ верны соотношения

$$\mathbf{P}(k) < \left[1 - \frac{1}{13}(k-1)p^2M^2\right]^m \leqslant e^{-(k-1)p^2M^2m/13} \leqslant n^{-k+1}. \tag{12}$$

Подставляя (10) и (12) в (3), находим

$$\mathbf{E}|T(n)| \leqslant 1 + n + \sum_{k=2}^{n}\binom{n}{k}(p(k,M))^m \leqslant 1 + n + \sum_{k=2}^{n}\binom{n}{k}n^{-k+1} = O(n).$$

Из вышеизложенного следует, что в формулировке теоремы 1 достаточно положить $c = 13$.

Проблемы точного оценивания константы c мы здесь не касаемся. Отметим лишь, что для случая $M = 1$, когда задача (1) является известной NP-полной задачей об упаковке [8, 21], подходит любое значение $c \geqslant 1$. Чтобы убедиться в этом, выпишем при $M = 1$ для вероятности $\mathbf{P}(k)$ оценку

$$\mathbf{P}(k) \leqslant (p(k,M))^m, \tag{13}$$

где

$$
\begin{aligned}
p(k,M) &= \mathbf{P}\{\sum_{i=1}^{k} a_{ij} \leqslant 1\} = (1-p)^k + kp(1-p)^{k-1} \\
&= (1-p)^{k-1}[1-p+kp] = (1-p)^{k-1}[1+p(k-1)] \\
&\leqslant (1-p)^{k-1}(1+p)^{k-1} = (1-p^2)^{k-1} \leqslant \exp(-p^2(k-1)).
\end{aligned}
\tag{14}
$$

Пользуясь (13) и (14), при $mp^2 \geqslant \ln n$ находим

$$
\begin{aligned}
\mathbf{E}|T(n)| &\leqslant 1+n+\sum_{k=2}^{n} \binom{n}{k} \exp(-mp^2(k-1)) \\
&\leqslant 1+n+\sum_{k=2}^{n} \binom{n}{k} n^{-k+1} = O(n).
\end{aligned}
$$

Согласно вышеизложенному мы фактически получили полиномиальный в среднем алгоритм для решения не только задачи (1), но и более трудной задачи подсчета числа целых точек в многограннике, задаваемом условием $A\mathbf{x} \leqslant \mathbf{b}$.

3. Трудность нетривиальной аппроксимации в худшем случае. Перейдем к анализу алгоритмов по худшему случаю и рассмотрим вопрос о существовании алгоритмов для приближенного решения задачи (1) за полиномиальное время, когда элементами матрицы A могут быть как положительные, так и отрицательные числа. Наша цель — показать, что даже когда нахождение какого-нибудь допустимого решения не представляет трудностей, найти нетривиальное приближение так же трудно, как и точное решение.

Будем говорить, что *задача* (1) *аппроксимируется с мультипликативной точностью* $f(L)$ *за полиномиальное время*, если существует алгоритм G такой, что для любого L и всех входов $\mathbf{c}$, $\mathbf{b}$, A с длиной двоичной записи L алгоритм G за полиномиальное время находит допустимое решение $\mathbf{x}_G = (x_1, \dots, x_n)$ такое, что $z_G \geqslant z/f(L)$, где $z_G = \sum_{i=1}^{n} c_i x_i$ и z — целочисленный оптимум задачи (1).

Рассмотрим задачу вида (1) с данными $\mathbf{c} \geqslant \mathbf{0}$, $\mathbf{b} \geqslant \mathbf{0}$, которая имеет ненулевое допустимое решение с одной единичной и остальными нулевыми компонентами. Тогда с мультипликативной точностью Mn задача (1) аппроксимируется тривиально. В настоящей работе получен отрицательный ответ на поставленный в [21] вопрос о возможности аппроксимации целочисленных оптимумов в задаче (1) некоторой функцией от исходных данных, вычислимой в полиномиальное время.

Теорема 2. *Если для некоторых* C, δ $(C \geqslant 1, 0 < \delta < 1)$ *задача* (1) *аппроксимируется с мультипликативной точностью* $C(n - n^\delta)$ *в полиномиальное время, то* $\mathcal{P} = \mathcal{NP}$.

Доказательство. Рассмотрим линейную задачу: найти максимум линейной формы

$$x_1 + Cx_{k+1} + \ldots + Cx_n,$$

где $x_1, \ldots, x_n$ — булевы переменные, при следующих ограничениях:

$$\sum_{j=1}^{k} a_{1j}x_j \leqslant 1, \ldots, \sum_{j=1}^{k} a_{kj}x_j \leqslant 1,$$

$$x_{k+1} - \sum_{j=1}^{k} a_{1j}x_j \leqslant 0, \ \ldots, \ x_{k+1} - \sum_{j=1}^{k} a_{kj}x_j \leqslant 0,$$

$$x_{k+2} - x_{k+1} \leqslant 0, \ldots, \ x_n - x_{n-1} \leqslant 0,$$

$$a_{ij} \in \{0, 1\}, \qquad i = 1, \ldots, k, \qquad j = 1, \ldots, k.$$

Положим $k = n^\delta$, где δ, $0 < \delta < 1$, — константа. Обозначим через $A' = (a_{ij})$ подматрицу матрицы A размеров $k \times k$, задаваемую первыми k неравенствами и первыми k переменными.

Оптимум z этой задачи больше единицы тогда и только тогда, когда A' содержит решение следующей известной задачи о покрытии: для данной $(0,1)$-матрицы A' определить, существует ли $(0,1)$-вектор $\mathbf{x}$ длины k такой, что $A'\mathbf{x} = \mathbf{1}$. Если точное покрытие для A' существует, то $z = C(n - n^\delta)$, в противном случае $z = 1$. Следовательно, если за полиномиальное время можно определить, будет ли z больше единицы, то за полиномиальное время решается задача о точном покрытии. Для проверки неравенства $z > 1$ достаточно аппроксимировать задачу (1) с точностью p, $p < C(n - n^\delta)$. Поскольку задача о точном покрытии является NP-полной [1], мы не можем аппроксимировать задачу (1) с точностью $C(n - n^\delta)$ в полиномиальное время, если только $\mathcal{P} \neq \mathcal{NP}$.

Отметим, что если каждый столбец матрицы A не превосходит $\mathbf{b}$ (т. е. векторы $\mathbf{e}^i$ допустимы при всех $i = 1, \ldots, n$), то с мультипликативной точностью n задача (1) аппроксимируется жадным алгоритмом, который из всех допустимых столбцов выбирает столбец с максимальным значением c_i до тех пор, пока остаются столбцы, которые можно добавить к выбранным без нарушения ограничений задачи (1).

Автор выражает глубокую благодарность А. Д. Коршунову за многочисленные замечания, способствовавшие улучшению первоначального варианта статьи. Автор благодарит также А. А. Сапоженко и В. К. Леонтьева за обсуждение полученных результатов и ряд ценных замечаний.

ЛИТЕРАТУРА

1. **Garey M. R., Johnson D. S.** Computers and intractability. San Fransisco: W. H. Freeman and Co., 1979.
2. **Gurevich Y.** Average case completeness // J. Comput. System Sci. 1991. V. 42, N 3. P. 346–398.
3. **Levin L.** Average case complete problems // SIAM J. Comput. 1986. V. 15, N 1. P. 285–286.
4. **Arora S., Lund C., Motwani R., Sudan M., Szegedy M.** Proof verification and hardness of approximation problems // Proc. 33rd annual sympos. on foundations of computer science. Los Alamitos: IEEE Computer Soc. Press, 1992. P. 14–23.
5. **Berman P., Schnitger G.** On the complexity of approximating the independent set problem // Inform. and Comput. 1992. V. 96, N 1. P. 77–94.
6. **Iwama K.** CNF satisfiability test by counting and polynomial average time // SIAM J. Comput. 1989. V. 18, N 2. P. 385–391.
7. **Lund C., Yannakakis M.** On the hardness of approximating minimization problems // Proc. of the 25th ACM sympos. on theory of computing. New York: ACM, 1993.
8. **Схрейвер А.** Теория линейного и целочисленного программирования. М.: Мир, 1991. Т. 2.
9. **Хачиян Л. Г.** Полиномиальный алгоритм в линейном программировании // Докл. АН СССР. 1979. Т. 244, № 5. С. 1093–1096.
10. **Klee V., Minty G. J.** How good is the simplex algorithm? //Inequalities, III. New York: Acad. Press, 1972. P. 159–175.
11. **Borgwardt K.-H.** The average number of pivot steps required by the simplex-method is polynomial // Z. Oper. Res. 1982. Ser. A–B. Bd. 26, N 5. S. 157–177.
12. **Smale S.** On the average number of steps of the simplex method of linear programming // Math. Programming. 1983. V. 27, N 3. P. 241–262.
13. **Brown C. A., Purdom P. W. (jr.)** An average time analysis of backtracking // SIAM J. Comput. 1981. V. 10, N 3. P. 583–593.
14. **Dinh Dieu P., Cong Thang L., Tuan Hoa L.** Average polynomial time complexity of some NP-complete problems // Theoret. Comput. Sci. 1986. V. 46, N 2. P. 219–237.
15. **Gurevitch Y., Shelah S.** Expected computation time for Hamiltonian path problem // SIAM J. Comput. 1987. V. 16, N 3. P. 486–502.
16. **Angluin D., Valiant L. G.** Fast probabilistic algorithms for Hamiltonian circuits and matchings // J. Comput. System Sci. 1979. V. 19, N 2. P. 155–193.

17. **Bollobas B., Fenner T. I., Frieze A. M.** An algorithm for finding Hamilton cycles in a random graph // Proc. of the 17th annual ACM sympos. on theory of computing. New York: ACM, 1985. P. 430–439.

18. **Lawler E. L.** Fast approximation algorithms for knapsack problems // Proc. 18th IEEE sympos. on foundation of computer science. New York: IEEE, 1977. P. 206–213.

19. **Ахо А., Хопкрофт Дж., Ульман Дж.** Построение и анализ вычислительных алгоритмов. М.: Мир, 1979.

20. **Гаврилов Г. П., Сапоженко А. А.** Задачи и упражнения по курсу дискретной математики. М.: Наука, 1992.

21. **Aharoni R., Erdős P., Linial N.** Optima of dual integer linear programs // Combinatorica. 1988. V. 8, N 1. P. 476–483.

Адрес автора:

РОССИЯ,
109004, Москва,
Б. Коммунистическая, 25
Институт системного программирования РАН

Статья поступила

23 мая 1994 г.