ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ
Январь—июнь 1999. Серия 2. Том 6, № 1, 33–42

УДК 519.83

# НОВЫЕ АЛГОРИТМЫ ОПТИМИЗАЦИИ В КОНУСЕ ЦЕНТРАЛЬНОГО ПУТИ*)

*В. И. Зоркальцев, А. Ю. Филатов*

Для решения задачи линейного программирования предложены новые алгоритмы оптимизации в конусе центрального пути. Приведены результаты численного эксперимента на тестовых примерах.

Рассмотрим взаимно двойственные задачи линейного программирования

$$c^T x \to \min_{x \in X}, \quad X = \{x \in R^n \mid Ax = b,\ x \geqslant 0\}, \tag{1}$$

$$b^T u \to \max_{u \in U}, \quad U = \{u \in R^m \mid g(u) \equiv c - A^T u \geqslant 0\}, \tag{2}$$

где $c \in R^n$; $b \in R^m$; $A$ — матрица размерности $m \times n$, $\operatorname{rank} A = m$; $X$, $U$ — множества допустимых решений задач (1) и (2).

Самосопряженная задача. Задачи (1), (2) эквивалентны одной задаче, двойственной самой себе:

$$f(x,u) = c^T x - b^T u = \sum_{j=1}^{n} x_j g_j(u) \to \min_{(x,u) \in D}. \tag{3}$$

Здесь $D = X \times U$ множество допустимых решений задачи (3). Через $X^*$, $U^*$ и $D^* = X^* \times U^*$ обозначим множества оптимальных решений задач (1)–(3).

Задача (3) обладает тем свойством, что $f(x,u) \geqslant 0$, $(x,u) \in D$, и $f(x^*,u^*) = 0$, $(x^*,u^*) \in D^*$. Последнее равенство влечет совпадение оптимумов задач (1) и (2): $c^T x^* = b^T u^*$ и выполнение условий дополняющей нежесткости:

$$x_j^* g_j(u^*) = 0, \quad j = 1, \ldots, n.$$

*) Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (код проекта 97-01-00859).

Будем предполагать, что множество $D$ непусто и имеет непустую относительную внутренность

$$\text{ri } D = \{(x,u) \in D \mid x_j > 0, \quad g_j(u) > 0, \quad j = 1,\dots,n\}.$$

Следовательно, $D^* \neq \varnothing$ и существует точка $(x^*,u^*) \in D^*$, в которой условия дополняющей нежесткости выполняются в строгой форме:

$$x_j^* g_j(u^*) = 0, \quad \max\{x_j^*, g_j(u^*)\} > 0, \quad j = 1,\dots,n.$$

Логарифмическая барьерная функция. Истоки алгоритмов центрального пути восходят к идее Фриша [6] включения в целевую функцию штрафных слагаемых в виде логарифмов невязок ограничений неравенств. Подобные конструкции активно использовались с конца 60-х годов в нелинейном программировании [5], однако лишь в последнее время были перенесены на линейные задачи.

Рассмотрим задачу минимизации логарифмической барьерной функции

$$f(x,u) - \mu \sum_{j=1}^{n} \ln(x_j g_j(u)) \to \min_{(x,u)\in D}. \tag{4}$$

Эта задача распадается на две:

$$c^T x - \mu \sum_{j=1}^{n} \ln x_j \to \min_{x} \qquad \text{при } Ax = b, \tag{5}$$

$$b^T u + \mu \sum_{j=1}^{n} \ln y_j \to \max_{(u,y)} \qquad \text{при } y + A^T u = c. \tag{6}$$

Множители Лагранжа ограничений одной задачи являются оптимальными значениями переменных другой задачи.

Центральный путь. Из предположения ri $D \neq \varnothing$ следует [4], что для любого $\mu > 0$ существует единственная пара $(x(\mu), u(\mu))$ векторов из $D$ такая, что для их компонент выполняются условия

$$x_j(\mu) g_j(u(\mu)) = \mu, \quad j = 1,\dots,n. \tag{7}$$

Точка $(x(\mu), u(\mu))$ является решением задачи (4). Множество $P = \{(x(\mu), u(\mu)) \mid \mu > 0\}$ назовем *центральным путем*. Из (7) следует, что при $\mu \to 0$ точки центрального пути сходятся к оптимальным решениям задачи (3).

Отметим, что точка $(x(\mu), u(\mu))$ является также решением задачи

$$\sum_{j=1}^{n} \ln x_j + \sum_{j=1}^{n} \ln g_j(u) \to \max_{(x,u)} \quad \text{при} \quad c^T x - b^T u = n\mu, \quad (x,u) \in D.$$

Следовательно, центральный путь состоит из точек поверхности $f(x,u) = n\mu$, наиболее удаленных в смысле суммы логарифмов от границы множества $D$.

Конус центрального пути. Расчет векторов $x(\mu)$, $u(\mu)$ требует большого объема вычислений. Поэтому при конструировании алгоритмов решения задач (1), (2) используют приближение к точкам центрального пути. Чтобы гарантировать сходимость к оптимальным решениям, точность такого приближения должна нарастать с уменьшением параметра $\mu$. Эти соображения отражены в следующем определении. Обозначим через $K(\theta,\mu)$ множество таких точек $(x,u) \in \text{ri } D$, что

$$\max_{j=1,\dots,n} |\mu - x_j g_j(u)| \leqslant \sqrt{\theta}\mu. \tag{8}$$

Множество $K(\theta) = \{(x,u) \in K(\theta,\mu) \mid \mu > 0\}$ назовем *конусом центрального пути* радиуса $\theta \geqslant 0$.

Очевидно, что $K(0) = P$ и $P \subset K(\theta)$. Пусть $0 < \theta < 1$. Если точка $(x,u)$ удовлетворяет (8), то $x_j g_j(u) > 0$ при всех $j = 1,\dots,n$. Поэтому включение $(x,u) \in \text{ri } D$ выполняется, если $Ax = b$ и все компоненты либо вектора $x$, либо вектора $g(u)$ положительны.

Суть алгоритмов оптимизации в конусе центрального пути заключается в построении такой последовательности точек $(x^k,u^k)$, $k = 0,1,2,\dots$, что $(x^k,u^k) \in K(\theta,\mu^k)$ и $\mu^{k+1} < \mu^k$. Если выполняется неравенство

$$\mu^{k+1} \leqslant (1 - \alpha/\sqrt{n})\mu^k,$$

то, как показано в [7, 8], исходная задача линейного программирования решается за $O(\sqrt{n}L)$ итераций, где $L$ — показатель «сложности» задачи, который при некоторых естественных условиях можно считать пропорциональным $\ln n$. Величина $\alpha$ близка к 0,35 и равна 0,125 для алгоритмов, предложенных соответственно в [7] и [8]. Рассматриваемые ниже алгоритмы имеют значение $\alpha$ не менее 0,5. Отметим, что в этих алгоритмах используется подмножество конуса $K(\theta)$, описываемое неравенством

$$\sum_{j=1}^{n} (\mu - x_j g_j(u))^p \leqslant (\sqrt{\theta}\mu)^p, \tag{9}$$

где $p$ — четное число. Очевидно, что из (9) следует (8).

**Алгоритм** $A$. Пусть имеется приближение $(x^k,u^k)$, $\mu^k$, удовлетворяющее условиям

$$(x^k, u^k) \in D, \quad \sum_{j=1}^{n} \left(\mu^k - x_j^k g_j(u^k)\right)^2 \leqslant \theta(\mu^k)^2.$$

Итеративный переход совершается по формулам

$$u^{k+1} = \arg\min_{u \in R^m} \sum_{j=1}^{n} \left(\mu^k - x_j^k g_j(u)\right)^2 = (AX_k^2 A^T)^{-1}(AX_k^2 c - \mu^k b),$$

$$x^{k+1} = 2x^k - X_k^2 g(u^{k+1})/\mu^k, \quad X_k^2 = \operatorname{diag}\{(x_j^k)^2\}, \tag{10}$$

$$\mu^{k+1} = (1-\beta)\mu^k, \quad \beta = \left(\sqrt{\theta(1-\theta)n} - \theta\right)\Big/(n-\theta). \tag{11}$$

Для вычисления $x^{k+1}$ здесь используется квадратичная аппроксимация задачи (5) при $x = x^k$ и $\mu = \mu^k$. Действительно, переход (10) можно представить в виде

$$x^{k+1} = x^k + s^k, \quad s_j^k = x_j^k(\mu^k - x_j^k g_j(u^{k+1}))/\mu^k, \quad j = 1, \ldots, n,$$

где вектор $s^k$ является решением задачи

$$c^T s - \mu^k \sum_{j=1}^{n} \left(\frac{s_j}{x_j^k} - \frac{1}{2}\left(\frac{s_j}{x_j^k}\right)^2\right) \to \min_s \quad \text{при} \quad As = 0.$$

Вектор $u^{k+1}$ состоит из множителей Лагранжа ограничений этой задачи.

**Алгоритм** $B$. Представляется разумным вместо (11) выбирать значение $\mu^{k+1}$, являющееся решением задачи

$$\mu \to \min \quad \text{при} \quad \sum_{j=1}^{n} \left(\mu - x_j^{k+1} g_j(u^{k+1})\right)^2 \leqslant \theta\mu^2.$$

Нетрудно видеть, что

$$\mu^{k+1} = (U - \sqrt{U^2 - V})/(n-\theta), \quad U = \sum_{j=1}^{n} x_j^{k+1} g_j(u^{k+1}),$$

$$V = (n-\theta)\sum_{j=1}^{n} \left(x_j^{k+1} g_j(u^{k+1})\right)^2.$$

При пересчете параметра $\mu$ по этому правилу выполняется неравенство $\mu^{k+1} < (1-\beta)\mu^k$. Для рассматриваемых ниже алгоритмов оно также справедливо, что позволяет считать эти алгоритмы улучшениями алгоритма $A$. Таким образом, объем вычислений, необходимый для решения задач (1), (2) алгоритмом $A$, служит верхней оценкой объема вычислений для алгоритма $B$ и последующих алгоритмов.

**Алгоритм** $C_p$. Рассмотрим параметризацию

$$u^k(\lambda) = \arg\min_{u\in R^m} \sum_{j=1}^{n} \Big(\lambda\mu^k - x_j^k g_j(u)\Big)^2 = (AX_k^2A^T)^{-1}(AX_k^2c - \lambda\mu^k b).$$

Обозначим через $\lambda^k$ решение задачи

$$\lambda \to \min \quad \text{при} \quad \sum_{j=1}^{n} \Big(\lambda\mu^k - x_j^k g_j(u^k(\lambda))\Big)^p \leqslant (\sqrt{\theta}\lambda\mu^k)^p,\ 0 \leqslant \lambda \leqslant 1.$$

Итеративный переход осуществляется по формулам

$$\mu^{k+1} = \lambda^k\mu^k, \quad u^{k+1} = u^k(\lambda^k),$$
$$x^{k+1} = x^k + s^k, \quad s_j^k = x_j^k\Big(\mu^{k+1} - x_j^k g_j(u^{k+1})\Big)\Big/\mu^{k+1}, \quad j = 1,\dots,n.$$

Алгоритмы $A$, $B$ и $C_2$ анонсированы в [2]. Их теоретическое обоснование приведено в [3]. При одних и тех же значениях $x^k$, $\mu^k$ выполняются неравенства

$$\mu_B^{k+1} \geqslant \mu_{C_2}^{k+1} \geqslant \mu_{C_4}^{k+1}. \tag{12}$$

Индексами $B$, $C_2$, $C_4$ обозначены значения $\mu^{k+1}$, вычисляемые соответствующими алгоритмами. Есть основания считать, что равенства в (12) реализуются редко.

Конечно, нельзя утверждать, что скорости сходимости алгоритмов должны распределяться согласно неравенствам (12). Они справедливы только при одном и том же исходном приближении на данной итерации. Вычислительные процессы по этим алгоритмам приводят к разным траекториям. Локальный выигрыш на одной итерации не означает ускорения процесса в целом. На одном из тестовых примеров подобная ситуация имела место. Однако, как показал эксперимент, она является скорее исключением, чем правилом.

**Алгоритм** $D_p$ развивает идеи алгоритма $C_p$. Пусть $\lambda^k$, $u^k(\lambda^k)$ те же величины, что и в алгоритме $C_p$. Для краткости записи введем обозначения $\tilde{\mu}^k = \lambda^k\mu^k$, $\tilde{u}^k = u^k(\lambda^k)$, $\tilde{g}^k = g(\tilde{u}^k)$.

Рассмотрим задачу

$$b^Tv + \lambda\tilde{\mu}^k \sum_{j=1}^{n} \left(\frac{z_j}{\tilde{g}_j^k} - \frac{1}{2}\Big(\frac{z_j}{\tilde{g}_j^k}\Big)^2\right) \to \max_{v,z} \quad \text{при} \quad z + A^Tv = 0, \tag{13}$$

которая является квадратичной аппроксимацией задачи (6) при $u = \tilde{u}^k$, $y = \tilde{g}^k$, $\mu = \lambda\tilde{\mu}^k$. Параметр $\lambda$ пробегает интервал $(0,1)$. Обозначим через $\tilde{v}^k(\lambda)$ решение задачи (13) и через $\tilde{x}^k(\lambda)$ — вектор множителей Лагранжа

ограничений (13). Положим $\tilde{r}^k(\lambda) = \lambda\tilde{\mu}^k\tilde{v}^k(\lambda)$. Функции $\tilde{r}^k(\lambda)$, $\tilde{x}^k(\lambda)$ линейны по $\lambda$ и имеют следующий вид:

$$\tilde{r}^k(\lambda) = (A\widetilde{G}_k^{-2}A^T)^{-1}(b - \lambda\tilde{\mu}^k A\widetilde{G}_k^{-1}e), \quad \tilde{x}^k(\lambda) = \lambda\tilde{\mu}^k\widetilde{G}_k^{-1}e + \widetilde{G}_k^{-2}A^T\tilde{r}^k(\lambda).$$

Здесь $\widetilde{G}_k^{-1} = \operatorname{diag}\{1/\tilde{g}_j^k\}$, $\widetilde{G}_k^{-2} = \operatorname{diag}\{1/(\tilde{g}_j^k)^2\}$, $e$ — вектор размерности $n$, все компоненты которого равны 1.

Обозначим через $\tilde{\lambda}^k$ решение задачи

$$\lambda \to \min \quad \text{при} \quad \sum_{j=1}^{n}\left(\lambda\tilde{\mu}^k - \tilde{x}_j^k(\lambda)\tilde{g}_j^k\right)^p \leqslant \left(\sqrt{\theta}\lambda\tilde{\mu}^k\right)^p, \quad 0 < \lambda < 1.$$

Итеративный переход осуществляется по формулам

$$\mu^{k+1} = \tilde{\lambda}^k\tilde{\mu}^k, \quad x^{k+1} = \tilde{x}^k(\tilde{\lambda}^k), \quad u^{k+1} = \tilde{u}^k + \tilde{r}^k(\tilde{\lambda}^k)/\mu^{k+1}.$$

В описываемом ниже численном эксперименте рассматривались алгоритмы $A$, $B$ и $C_p$, $D_p$ при $p = 2, 4, 8$. Для сравнения использовался следующий вариант афинного метода [1], не являющегося полиномиальным, но хорошо показавшим себя при решении практических задач.

**Алгоритм** $E$. Итеративный переход совершается по правилу

$$u^{k+1} = (AX_k^2A^T)^{-1}AX_k^2c, \quad x^{k+1} = x^k + \lambda^k s^k,$$

где $X_k^2 = \operatorname{diag}\{(x_j^k)^2\}$. Направление корректировки $s^k$ является решением вспомогательной задачи

$$c^Ts + \frac{1}{2}\sum_{j=1}^{n}\left(\frac{s_j}{x_j^k}\right)^2 \to \min_s \quad \text{при} \quad As = 0$$

и выражается формулой $s^k = -X_k^2g(u^{k+1})$. Шаг $\lambda^k$ должен сохранять положительность компонент $x_j^{k+1}$. В расчетах он вычислялся по формуле

$$\lambda^k = 0,6 \times \min\{-x_j^k/s_j^k \mid j = 1, \ldots, n,\ s_j^k < 0\}.$$

Инициализация алгоритмов. Приведем один из способов построения начального приближения $(x^0, u^0)$, $\mu^0$, предложенный в [8]. Обозначим через $M$ максимум среди абсолютных значений определителей всех квадратных подматриц матрицы $A$ и через

$$L = \ln(1 + M) + \ln\left(1 + \max_j|c_j|\right) + \ln\left(1 + \max_i|b_i|\right) + \ln(m + n)$$

показатель сложности задачи (1). Положим $d = 4^L$ и $h = d^2$.

Вместо задачи (1) будем рассматривать расширенную задачу размерности $(m+1)\times(n+2)$, которая имеет матрицу ограничений $\tilde{A}$ и векторы $\tilde{b}$, $\tilde{c}$ следующего вида:

$$\tilde{A}=\begin{pmatrix} a_{11} & \dots & a_{1n} & 0 & b_1-d\sum a_{1j} \\ \dots & \dots & \dots & \dots & \dots \\ a_{m1} & \dots & a_{mn} & 0 & b_m-d\sum a_{mj} \\ h-c_1 & \dots & h-c_n & h & 0 \end{pmatrix},\quad \tilde{b}=\begin{pmatrix} b_1 \\ \dots \\ b_m \\ b_{m+1} \end{pmatrix},$$

$$\tilde{c}^T=(c_1\ \dots\ c_n\ 0\ dh),$$

где $b_{m+1}=dh(n+1)-d\sum c_j$. Обозначим через $(x^*,u^*)$ оптимальное решение соответствующей задачи (3). В [8] доказано, что

а) если $x^*_{n+2}=g_{n+1}(u^*)=0$, то решения исходной и расширенной задач совпадают;

б) если $x^*_{n+2}\neq 0$, то ограничения исходной задачи несовместны;

в) если $g_{n+1}(u^*)\neq 0$, то целевая функция исходной задачи неограничена.

Рассмотрим приближение

$$x^0=(d,\dots,d,1),\quad u^0=(0,\dots,0,-1),\quad \mu^0=dh.$$

Нетрудно проверить, что точка $(x^0,u^0)$ допустима в задаче (3) и справедливы равенства $x^0_j g_j(u^0)=\mu^0$, $j=1,\dots,n+2$. Следовательно, точка $(x^0,u^0)$ лежит на центральном пути при $\mu=\mu^0$. Недостатком данного способа инициализации является сильная «перекошенность» расширенной задачи ввиду больших значений $d$ и $h$. Можно несколько сократить этот эффект, задавая меньшие значения $d$ и $h$, что и было сделано при решении тестовых задач.

Тестовые задачи. Ниже приведены задачи 1–5, на которых проверялась работоспособность алгоритмов. Задачи 1–3 представлены матрицами $A$ и векторами $b$, $c$. Задачи 4 и 5 записаны в виде (2). Размерности указаны для расширенных задач.

**Задача 1** $(2\times 4)$: $A=(1\ 1)$, $b=(1)$, $c^T=(1\ 2)$.

**Задача 2** $(3\times 6)$: $A=\begin{pmatrix}5&3&1&0\\3&2&0&1\end{pmatrix}$, $b=\begin{pmatrix}480\\300\end{pmatrix}$, $c^T=(-1.2,\ -1,\ 0,\ 0)$.

**Задача 3** $(8\times 18)$:

$$A=\begin{pmatrix}
15&0&0&0&30&0&0&0&25&0&0&0&-1&0&0&0\\
0&10&0&0&0&25&0&0&0&50&0&0&0&-1&0&0\\
0&0&20&0&0&0&10&0&0&0&30&0&0&0&-1&0\\
0&0&0&50&0&0&0&17&0&0&0&45&0&0&0&-1\\
1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0
\end{pmatrix},$$

$$b = \begin{pmatrix} 300 \\ 200 \\ 1000 \\ 500 \\ 50 \\ 20 \\ 30 \end{pmatrix}, \quad c^T = (15\ 20\ 25\ 40\ 70\ 28\ 15\ 45\ 40\ 70\ 40\ 65\ 0\ 0\ 0\ 0).$$

**Задача 4** ($(m+1) \times (2m+2)$): $u_m \to \max$, $|u_1| \leqslant 1$, $|u_i| \leqslant 1 + u_{i-1}$, $i = 2, \ldots, m$. Оптимальное решение: $u_i^* = i$, $i = 1, \ldots, m$. Задача 4 решалась при $m = 18$ и $m = 99$.

**Задача 5** ($(m+1)\times(2m+2)$): $u_m \to \max$, $|u_1| \leqslant 1$, $|u_i| \leqslant 2^{2i-2} + 2u_{i-1}$, $i = 2, \ldots, m$. Оптимальное решение: $u_1^* = 1$, $u_i^* = 2^{2i-2} + 2u_{i-1}^*$, $i = 2, \ldots, m$. Задача 5 решалась при $m = 5$ и $m = 18$.

Численный эксперимент (результаты расчетов приведены в таблице). Основной целью являлось сопоставление предложенных алгоритмов на тестовых задачах. Итеративный процесс начинался с одной точки, полученной в соответствии с процедурой инициализации. Значения $d$ для рассматриваемых задач принимались равными 1, 256, 256, $(1,2)^{18}$, $10^6$, $10^6$, $(1,2)^{100}$. Критерием останова служило достижение суммой $\sum x_j g_j$ значения $5\times10^{-6}$ во всех задачах, кроме задачи 5 размерности $19 \times 38$, в которой сумма $\sum x_j g_j$ на выходе составляла $10^{-3}$. Радиус $\theta$ конуса центрального пути принимался для алгоритма $A$ равным 0,5. Для других алгоритмов через дробь записаны числа итераций при радиусах $\theta = 0,5$ и $\theta = 0,9$. В задаче 4 размерности $100 \times 200$ значение $\theta$ равно 0,9 для всех алгоритмов, кроме $A$.

*Число итераций, необходимое для решения задач различными методами*

| Алгоритм | Задача 1 2 × 4 | Задача 2 3 × 6 | Задача 5 6 × 12 | Задача 3 8 × 18 | Задача 5 19 × 38 | Задача 4 19 × 38 | Задача 4 100 × 200 |
|---|---|---|---|---|---|---|---|
| $A$ | 90 | 218 | 278 | 258 | 730 | 805 | 2174 |
| $B$ | 38/30 | 113/87 | 158/119 | 154/117 | 459/342 | 507/380 | 1093 |
| $C_2$ | 28/20 | 50/37 | 85/63 | 82/59 | 260/189 | 240/170 | 501 |
| $D_2$ | **16/12** | **42/33** | **57/44** | **55/43** | **157/120** | **123/95** | **193** |
| $C_4$ | 25/17 | 45/29 | 62/45 | 50/42 | 119/101 | 110/79 | 142 |
| $D_4$ | **14/11** | **34/27** | **39/30** | **34/26** | **82/64** | **79/61** | **95** |
| $C_8$ | 24/12 | 45/32 | 57/39 | 46/33 | 105/78 | 89/63 | 87 |
| $D_8$ | 13/11 | 31/24 | 33/25 | 27/21 | 62/48 | 64/49 | 68 |
| $E$ | 16 | 35 | 36 | 30 | 62 | 66 | 84 |

В таблице выделены строки, соответствующие обоснованным алгоритмам, показавшим на тестовых примерах наилучшие результаты. Подчеркнутые значения показывают, что при ускорении сходимости к нулю параметра центрального пути на каждой отдельной итерации не всегда ускоряется процесс в целом.

Все исследуемые алгоритмы имеют примерно одинаковый объем вычислений на каждой итерации, что позволяет использовать число итераций для их сравнения. В результате эксперимента можно сделать вывод о том, что лучшие из алгоритмов оптимизации в конусе центрального пути не уступают методам, применяющимся на практике. Их перспективность повышает наличие полиномиальной верхней оценки числа итераций. Фактически оно существенно меньше теоретических оценок, что следует из значительного (до десятков раз) превосходства наиболее эффективных алгоритмов над алгоритмом $A$, число итераций которого близко к теоретическому.

Хотя наилучшие теоретические оценки получаются при значении радиуса $\theta = 0,5$, доставляющем максимум произведению $\theta(1-\theta)$, на практике оказываются эффективнее алгоритмы с более широкими аппроксимациями конуса центрального пути. Это достигается увеличением его радиуса $\theta$ и использованием во вспомогательных задачах высоких степеней $p$ вплоть до применения чебышевской нормы, что соответствует исходному определению конуса центрального пути.

## ЛИТЕРАТУРА

1. **Дикин И. И., Зоркальцев В. И.** Итеративное решение задач математического программирования: алгоритмы метода внутренних точек. Новосибирск: Наука, 1980.
2. **Зоркальцев В. И.** Метод наименьших квадратов: геометрические свойства, альтернативные подходы, приложения. Новосибирск: Наука, 1995.
3. **Зоркальцев В. И., Нечаева М. С.** Оптимизация в конусе центрального пути. Иркутск, 1995. (Препринт / РАН. Сиб. отд-ние. СЭИ; № 2).
4. **Зоркальцев В. И., Филатов А. Ю.** Исследование алгоритмов оптимизации в конусе центрального пути. Иркутск, 1997. (Препринт / РАН. Сиб. отд-ние. СЭИ; № 7).
5. **Фиакко А., Мак-Кормик Г.** Нелинейное программирование. Методы последовательной безусловной минимизации. М.: Мир, 1972.
6. **Frisch K.** The logarithmic potential method for convex programming. Oslo: Inst. of Economics, 1955.

7. **Kojima M., Mizuno S., Yoshise A.** A polynomial-time algorithm for a class of linear complementarity problems // Math. Programming. Ser. A. 1989. V. 44, N 1. P. 1–26.

8. **Monteiro R., Adler I.** Interior path following primal-dual algorithms. Pt 1: Linear programming // Math. Programming. Ser. A. 1989. V. 44, N 1. P. 27–41.

Адрес авторов:

ИСЭМ СО РАН,
ул. Лермонтова, 130,
664033 Иркутск, Россия.
E-mail: fial@isem.sei.irk.ru

Статья поступила
31 августа 1998 г.