ДИСКРЕТНЫЙ АНАЛИЗ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ
Июль—декабрь 2001. Серия 2. Том 8, № 2, 52–72

УДК 519.852

НОВЫЙ ПРИБЛИЖЕННЫЙ АЛГОРИТМ ДЛЯ РЕШЕНИЯ ЗАДАЧИ ПОЛОЖИТЕЛЬНОГО ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ*)

С. А. Фомин

Предлагается новый алгоритм нахождения ε-оптимального решения задачи положительного линейного программирования: найти $\max\{cx \mid Ax \leqslant b, x \geqslant 0\}$, где все исходные данные неотрицательны. Алгоритм не уступает лучшим из известных алгоритмов для этой задачи по верхним оценкам времени выполнения, в частности для него доказана верхняя оценка вычислительной сложности $O(\frac{mn}{\varepsilon^2}\log^2\frac{mn}{\varepsilon})$, и имеет простые последовательные и параллельные реализации. Проведенные вычислительные эксперименты показывают преимущество нового алгоритма над алгоритмами, разработанными в последние годы [3, 4].

Введение

В настоящей статье рассматриваются задачи положительного линейного программирования (ПЛП), также называемые задачами дробной упаковки, когда требуется найти вектор $x = (x_1, \ldots, x_n)$ такой, что

$$cx \to \max, \quad Ax \leqslant b, \quad x_j \geqslant 0,$$

где $A = (a_{ij})$ — матрица размера $m \times n$, a_{ij} — неотрицательные рациональные числа; $c = (c_1, \ldots, c_m)$ — рациональный вектор, $c_j \geqslant 0$; $b = (b_1, \ldots, b_m)$ — рациональный вектор, $b_i \geqslant 0$, и задачи дробного покрытия (когда линейные ограничения есть ограничения вида $Ax \geqslant b$, причем все данные неотрицательны, и необходимо минимизировать целевую функцию).

Многие задачи оптимизации можно представить в виде задачи ПЛП, в частности задачи оптимизации потоков в сетях [3, 4]. В [10] была доказана P-полнота задачи ПЛП. Это означает, что любую полиномиально разрешимую задачу можно LOGSPACE свести к задаче ПЛП.

*) Исследование выполнено при финансовой поддержке Российского фонда фундаментальных исследований (проект 99-01-00210).

Несмотря на существование полиномиальных алгоритмов решения задач линейного программирования [2, 6], рост размерности задачи, коммуникационные аспекты, возможность параллельной реализации подталкивают к поиску приближенных алгоритмов для решения задач ПЛП. Заметим, что при доминирующей гипотезе о различии классов NC и P из P-полноты задачи ПЛП следует отсутствие эффективных параллельных алгоритмов точного решения задачи ПЛП. С другой стороны, в последние годы было разработано много ε-приближенных ПЛП-алгоритмов [3–5, 7, 9]. Однако заметим, что за исключением работы [1], где исследовалась реализация алгоритма из [3], не было исследований реализаций этих алгоритмов.

В данной статье предлагается новый алгоритм для решения задачи ПЛП, который по оценкам вычислительной сложности не уступает существующим алгоритмам [3, 4, 7] и имеет простую эффективную последовательную и параллельную реализации. Для предложенного алгоритма доказано, что его вычислительная сложность не превосходит $O(\frac{mn}{\varepsilon^2}\log^2\frac{mn}{\varepsilon})$. Были проведены исследования эффективности реализаций предлагаемого алгоритма и алгоритмов, разработанных в последние годы [3, 4]. На основании вычислительных экспериментов можно говорить о преимуществе предлагаемого алгоритма над алгоритмами из [3, 4]; при этом верхние оценки вычислительной сложности нового алгоритма лучше оценок для алгоритма из [4] и не хуже оценок для алгоритма из [3].

В разд. 2, 3 приведены сравнение и анализ упомянутых алгоритмов с позиции вычислительной сложности, а в разд. 4 — результаты исследования программных реализаций.

1. Постановка задачи ПЛП

Ниже n обозначает число переменных и m — число ограничений прямой задачи (не считая ограничений вида $x_i \geqslant 0$).

Будем использовать j для индексирования переменных прямой задачи и i для индексирования ограничений. Где не будет явно указано обратное, будем считать, что j изменяется в интервале $[1, n]$, а i — в интервале $[1, m]$.

Для $x = (x_1, \ldots, x_n)$ определим $X = \sum\limits_j x_j$, а для $y = (y_1, \ldots, y_m)$ определим $Y = \sum\limits_i y_i$. Рассмотрим ПЛП-задачу в следующей *стандартной форме.*

Прямая задача ПЛП. Найти такой вектор $x = (x_1, \ldots, x_n)$, на котором достигается максимальное значение X и который удовлетворяет

следующим ограничениям: $a_{ij} \geqslant 0$, $x_j \geqslant 0$ при любых i, j и

$$\sum_j a_{ij} x_j \leqslant 1.$$

Двойственная задача ПЛП. Найти такой вектор $y = (y_1, \ldots, y_m)$, на котором достигается минимальное значение Y и который удовлетворяет следующим ограничениям: $a_{ij} \geqslant 0$ и $x_j \geqslant 0$ при любых i, j и

$$\sum_i a_{ij} y_i \geqslant 1.$$

Сразу заметим, что ПЛП-задача: максимизировать

$$\sum_{j=1}^{n} C_j x_j$$

при ограничениях

$$\sum_j d_{i,j} x_j \leqslant B_i, \quad d_{i,j} \geqslant 0, \tag{1}$$

при любых i, j преобразуется в стандартную форму посредством преобразования матрицы ограничений:

$$a_{ij} = \frac{d_{ij}}{B_i C_j}. \tag{2}$$

Очевидно, что такое преобразование не влияет на значение целевой функции оптимального решения и оптимальное решение задачи в стандартной форме, полученной из (1) посредством преобразования (2), будучи подвергнуто преобразованию, обратному (2), будет оптимальным решением задачи в исходной форме (1).

Ниже покажем, что без потери общности можно считать, что задача ПЛП подается на вход в *специальной форме*, в которой ненулевые коэффициенты a_{ij} матрицы ограничений удовлетворяют условиям $\frac{1}{\gamma} \leqslant a_{ij} \leqslant 1$, где $\gamma = \frac{4\kappa^2 mn}{\varepsilon^2}$. Далее, при описании предлагаемого алгоритма будем подразумевать, что входная ПЛП-задача приведена к специальной форме (для данного ε — параметра мультипликативной точности требуемого решения и данного κ).

Покажем, как произвести округление входной задачи ПЛП в стандартной форме, чтобы привести его к специальной форме (для данных ε и κ), причем значение решения ПЛП-задачи в специальной форме должно быть не меньше значения решения ПЛП-задачи в стандартной форме, умноженного на $(1 - \frac{\varepsilon}{\kappa})$.

Пусть $\beta = \min\limits_j \max\limits_i a_{ij}$ и $\mathrm{opt} = X^* = \sum\limits_j x_j^* = Y^* = \sum\limits_i y_i^*$ — значение оптимального решения ПЛП-задачи в стандартной форме. Тогда справедлива

Лемма 1.

$$\frac{1}{\beta} \leqslant \mathrm{opt} \leqslant \frac{m}{\beta}.$$

Доказательство. Первое неравенство следует из допустимости решения прямой ПЛП-задачи в стандартной форме:

$$k = \arg\min_j \max_i a_{ij}, \quad x_k = 1/\beta \ \text{ и } \ x_j = 0 \ \text{ при любом } \ j \neq k.$$

Второе неравенство следует из допустимости решения $y_i = 1/\beta$ для любого i двойственной ПЛП-задачи в стандартной форме. Лемма 1 доказана.

Теперь произведем округление стандартной формы (данное округление близко к округлениям из работ [3, 7]) и полученную форму назовем *округленной*. Для этого выполним следующие преобразования ненулевых коэффициентов $a_{ij} = 0$ матрицы (для заданных $0 < \varepsilon \leqslant 1$ и $\kappa > 2$):

$$\text{если } \ a_{ij} < \frac{\varepsilon\beta}{2\kappa m}, \ \text{ то } \ a'_{ij} = \frac{\varepsilon\beta}{2\kappa m}; \tag{3}$$

$$\text{если } \ a_{ij} > \frac{2\kappa n\beta}{\varepsilon}, \ \text{ то } \ x_j \equiv 0. \tag{4}$$

В преобразовании (4) запись $x_j \equiv 0$ означает, что для оптимального решения полагаем $x_j = 0$ и исключаем j-й столбец из матрицы ограничений. Легко видеть, что любое допустимое решение округленной формы x' является также допустимым решением стандартной формы. Действительно, в результате округления все неравенства могут только усилиться. Более формально, пусть при любом i

$$U_i = \left\{ j \mid a_{ij} > \frac{2\kappa n\beta}{\varepsilon} \right\}, \quad L_i = \left\{ j \mid a_{ij} < \frac{\varepsilon\beta}{2\kappa m} \right\},$$
$$N_i = \left\{ j \mid \frac{\varepsilon\beta}{2\kappa m} \leqslant a_{ij} \leqslant \frac{2\kappa n\beta}{\varepsilon} \right\}.$$

Тогда при любом i

$$\sum_j a_{ij}x'_j = \sum_{j\in L_i} a_{ij}x'_j + \sum_{j\in U_i} a_{ij}x'_j + \sum_{j\in N_i} a_{ij}x'_j$$
$$= \sum_{j\in L_i} a_{ij}x'_j + \sum_{j\in N_i} a_{ij}x'_j \leqslant \sum_{j\in L_i} a'_{ij}x'_j + \sum_{j\in N_i} a'_{ij}x'_j \leqslant \sum_j a'_{ij}x'_j \leqslant 1.$$

Также заметим, что так как указанные выше преобразования округления не изменяют β, то лемма 1 будет справедлива после преобразований (3) и (4). Действительно, преобразование (3) не изменяет значения β, так как $\varepsilon \leqslant 1$, $k \geqslant 1$, $m \geqslant 1$ и $\frac{\varepsilon\beta}{2\kappa m} < \beta \leqslant \max_i a_{ij}$, и после преобразования (3) при любом j значение $\max\limits_i a_{ij}$ не изменится, соответственно

не изменится и β. Аналогично после преобразования (4) удаляются только те столбцы j, для которых

$$\max_i a_{ij} \geqslant \frac{2\kappa n\beta}{\varepsilon} > \beta = \min_j \max_i a_{ij}.$$

Теперь покажем, что приведенное округление приведет к потере точности оптимального решения не более чем в $(1 - \frac{\varepsilon}{\kappa})$ раз. Пусть $a_{ij}^{(1)}$ — коэффициенты, полученные после применения преобразования (3) к коэффициентам матрицы исходной задачи в стандартной форме, PLP1 — соответствующая коэффициентам ПЛП-задача, $x^{(1)}$ — соответствующее оптимальное решение задачи PLP1. Пусть $x^{(1)}$ — оптимальное решение исходной задачи в стандартной форме. Тогда $x(1 - \frac{\varepsilon}{2\kappa})$ — допустимое решение задачи PLP1. Действительно, используя лемму 1, при любом i получаем

$$\sum_j a_{ij}^{(1)} x_j \Big(1 - \frac{\varepsilon}{2\kappa}\Big) \leqslant \Big(1 - \frac{\varepsilon}{2\kappa}\Big) \sum_j \Big(a_{ij} + \frac{\varepsilon\beta}{2\kappa m}\Big) x_j$$

$$\leqslant \Big(1 - \frac{\varepsilon}{2\kappa}\Big)\Big(\sum_j a_{ij}x_j + \frac{\varepsilon\beta}{2\kappa m}\sum_j x_j\Big) \leqslant \Big(1 - \frac{\varepsilon}{2\kappa}\Big)\Big(1 + \frac{\varepsilon\beta}{2\kappa m} \cdot \frac{m}{\beta}\Big)$$

$$\leqslant \Big(1 - \frac{\varepsilon}{2\kappa}\Big)\Big(1 + \frac{\varepsilon}{2\kappa}\Big) = 1 - \frac{\varepsilon^2}{4\kappa^2} \leqslant 1.$$

Следовательно,

$$\sum_j x_j^{(1)} \geqslant \sum_j x_j \Big(1 - \frac{\varepsilon}{2\kappa}\Big).$$

Пусть теперь PLP2 — задача, полученная из задачи PLP1 после преобразования (4), $a_{ij}^{(2)}$ — соответствующие коэффициенты, $x^{(2)}$ — соответствующее оптимальное решение. Пусть $U = \{j \mid$ существует i такое, что $a_{ij} > \frac{2\kappa n\beta}{\varepsilon}\}$. Тогда

$$\sum_j x_j^{(2)} = \sum_{j \notin U} x_j^{(2)} \geqslant \sum_{j \notin U} x_j^{(1)} \geqslant \sum_j x_j^{(1)} - |U| \max_{j \in U} x_j^{(1)}$$

$$\geqslant \sum_j x_j^{(1)} - n \cdot \frac{\varepsilon}{2\kappa n\beta} \geqslant \sum_j x_j^{(1)} - \frac{\varepsilon}{2\kappa\beta} \geqslant \sum_j x_j^{(1)} \Big(1 - \frac{\varepsilon}{2\kappa}\Big) \geqslant \sum_j x_j \Big(1 - \frac{\varepsilon}{\kappa}\Big).$$

Ясно, что отношение значения оптимального решения задачи PLP2, полученной из ПЛП-задачи в форме (1), к значению оптимального решения исходной ПЛП-задачи в форме (1) не меньше $(1 - \frac{\varepsilon}{\kappa})$, причем оптимальное решение задачи PLP2 будет допустимым решением задачи в форме (1). Теперь проведем масштабирование задачи PLP2. Пусть $a_{\max} = \max_{ij} a_{ij}^{(2)}$. Выполнив следующее преобразование ненулевых коэффициентов матрицы ограничений

$$a_{ij}^{s} = a_{ij}^{(2)} / a_{\max},$$

перейдем к *специальной* форме задачи ПЛП, в которой ненулевые коэффициенты a_{ij}^s матрицы ограничений удовлетворяют условию $\frac{1}{\gamma} \leqslant a_{ij}^s \leqslant 1$, где $\gamma = \frac{4\kappa^2 mn}{\varepsilon^2}$. Оптимальное решение задачи в специальной форме, деленное на $a_{\max}$, будет допустимым решением округленной и исходной форм, причем отношение значения оптимального решения задачи, нормированного на $a_{\max}$, к значению оптимального решения исходной ПЛП-задачи в форме (1) не меньше $(1 - \frac{\varepsilon}{\kappa})$.

2. Обзор алгоритмов для приближенного решения задачи ПЛП

Приближенный алгоритм называется *алгоритмом с мультипликативной точностью* D, если при любых исходных данных этот алгоритм находит допустимое решение со значением целевой функции, отличающемся от оптимума не более чем в D раз. Алгоритм с мультипликативной точностью $1 + \varepsilon$ называется *ε-оптимальным* (подразумевается, что ε близко к нулю). Термин *ε-оптимальное решение* используется для обозначения допустимого решения со значением целевой функции, отличающегося от оптимума не более чем в $(1 + \varepsilon)$ раз.

В [7] был описан ε-оптимальный алгоритм для прямой и двойственной ПЛП-задач со следующими свойствами.

Получив на вход описание задачи и параметр $\varepsilon > 0$, алгоритм находит такие допустимые решения x и y прямой и двойственной задач соответственно, что

$$\sum_i y_i \leqslant \sum_j x_j(1 + \varepsilon).$$

Этот алгоритм использует $O(\varepsilon^{-4} \log n \log(m/\varepsilon))$ итераций. Алгоритм допускает параллельную реализацию, так как каждая итерация может быть выполнена с вычислительной сложностью $O(\log N)$ параллельно на машине типа EREW PRAM с использованием $O(N)$ процессоров, где N есть число таких пар (i, j), что $a_{i,j} > 0$.

Затем в [3] был описан алгоритм для приближенного решения прямой и двойственной ПЛП-задач со следующими свойствами.

Получив на вход описание задачи и параметры $\varepsilon > 0$ и $0 < r < \ln(m^3\varepsilon^{-2})$, алгоритм находит такие допустимые решения x и y прямой и двойственной задач соответственно, что

$$\sum_i y_i \leqslant \sum_j x_j(r + (1 + \varepsilon)^2).$$

В [3] было показано, что число итераций ($\gamma = m^2\varepsilon^{-2}$) не превосходит

$$O(r^{-1}\varepsilon^{-2} \log^2(\gamma m) \log(\gamma mn/\varepsilon)).$$

При $r \approx \varepsilon$ эта оценка принимает вид $O(\varepsilon^{-3}\log^3(mn/\varepsilon))$. К сожалению, при анализе времени выполнения, приведенном в [3], нами обнаружена ошибка. На самом деле (см. разд. 6) правильная оценка числа итераций не превосходит

$$O(r^{-2}\varepsilon^{-2}\log(\gamma m)^2\log(\gamma mn/\varepsilon))$$

или (при $r \approx \varepsilon$) $O(\varepsilon^{-4}\log^3(mn/\varepsilon))$, что асимптотически эквивалентно числу итераций алгоритма из [7]. Алгоритм из [3] также допускает параллельную реализацию, однако исследования последовательной реализации на доступных (для имеющихся вычислительных ресурсов) объемах данных показали, что на этих данных алгоритм существенно уступает традиционным алгоритмам точного решения задачи ЛП, таким как симплекс-метод и метод внутренней точки.

В [4] был представлен ε-оптимальный алгоритм решения задачи ПЛП, использующий $O(\varepsilon^{-2}m\log m)$ итераций и $O(N)$ арифметических операций, (напомним, что N обозначает число ненулевых элементов в матрице ограничений задачи ПЛП). Легко видеть, что данный алгоритм в асимптотике уступает по числу итераций алгоритмам из [3, 7], имеющим при фиксированном ε субполиномиальные верхние оценки числа итераций. При фиксированном ε алгоритмы из [3, 7] имеют полилогарифмические верхние оценки числа итераций, а алгоритм из [4] — полиномиальную верхнюю оценку, причем существуют входные данные, на которых эта оценка достигается.

Однако исследование его реализации показало, что за счет выбора эвристик и простоты алгоритм демонстрировал приемлемое (порядка времени работы симплекс-метода) время работы даже для небольших квадратных матриц порядка 100.

Таким образом, интересно получить простой и быстрый алгоритм приближенного решения задачи ПЛП, не уступающий существующим алгоритмам по вычислительной сложности и допускающий параллельную реализацию. Такой алгоритм описывается в следующем разделе.

3. Новый алгоритм для приближенного решения задачи ПЛП

Формальное псевдоалгоритмическое описание алгоритма приведено в табл. 1. Заметим, что идея алгоритма близка к идеям алгоритмов из [3, 4, 7]. Она основана на последовательном увеличении переменных прямой задачи в зависимости от переменных двойственной задачи, соответствующих ограничениям прямой задачи.

Получив на вход матрицу ограничений ПЛП-задачи в специальной форме, алгоритм инициализирует наборы переменных, соответствующих переменным $x_1, \ldots x_n$ прямой задачи, переменным $y_1, \ldots y_n$

Таблица 1

Новый алгоритм для приближенного решения задачи ПЛП

procedure Initialize()

$\{\xi = \frac{\epsilon}{3}; \quad \mu = \frac{\epsilon}{3}; \quad \phi = \frac{(1+\epsilon)(2\ln m+\xi)}{(1+\epsilon)-(1+\xi)(1+\mu)}; \quad \psi = \phi - \xi;$

$\forall i \quad \hat{\lambda}_i = \sum_j a_{ij}; \qquad \hat{\lambda}_{\max} = \max_i \hat{\lambda}_i;$

$\forall j \quad x_j = x^0 = \frac{\ln m}{\phi \hat{\lambda}_{\max}};$

$\forall i \quad \lambda_i = \lambda_i^0 = \sum_j a_{ij} x_j; \quad y_i = e^{\phi\lambda_i - \psi};$

$\forall j \quad \alpha_j = \alpha_j^0 = \sum_i a_{ij} y_i;$

$\lambda_{\max} = \max_i \lambda_i; \quad X = 0; \quad Y = Y^0 = \sum_i y_i; \quad Y_{\min} = \infty\};$

procedure Iteration()

$\{Q = \{j : \alpha_j \leqslant \widehat{\alpha} \cdot (1+\mu)\}\};$

for $(j \in Q)$

$\{\Delta x_j = \frac{\xi}{\phi\lambda_{\max}} x_j; \quad x_j = x_j + \Delta x_j; \qquad X = X + \Delta x_j;$

$\forall i \quad \lambda_i = \lambda_i + a_{ij} \cdot \Delta x_j; \qquad q = j\};$

$\forall i \quad y_i = e^{\phi\lambda_i - \psi}; \qquad Y = \sum_i y_i; \quad \lambda_{\max} = \max_i \lambda_i;$

$\forall j \in Q; \quad \alpha_j = \sum_i a_{ij} y_i; \qquad Y_{\min} = \min\left(Y_{\min}, \frac{Y}{\min_j \alpha_j}\right)\};$

procedure Phase() $\{\widehat{\alpha} = \min_j \alpha_j;$

do { **call Iteration()** };

until $(\mathbf{Q} = \varnothing)$;

$\forall j \in \{j : \alpha_j \leqslant \alpha_q \cdot (1+\mu)\} \quad \alpha_j = \sum_i a_{ij} y_i\};$

Algorithm() { **call Initialize()**;

repeat until $\left((Y < 1) or \left(\frac{Y_{\min}\lambda_{\max}}{X} < (1+\epsilon)\right)\right)$

{ **call Phase()** };

$\forall j \quad x_j = x_j - x^0;$

$\forall i \quad \lambda_i = \sum_j a_{ij} x_j; \qquad \lambda_{\max} = \max_i \lambda_i;$

$\forall j \quad$ **output** $\frac{x_j}{\lambda_{\max}}\};$

двойственной задачи, значениям левых частей неравенств для $\lambda_1 \dots \lambda_m$ прямой задачи, для $\alpha_1 \dots \alpha_n$ двойственной задачи, а также переменные, определяющие зависимость переменных двойственной задачи от переменных прямой задачи. Выполнение алгоритма подразделяется на фазы, соответствующие процедуре **Phase** в псевдоалгоритмическом описании, включающие в себя ненулевое число итераций, соответствующих процедуре **Iteration**. Внутри каждой фазы по значениям переменных $\alpha_1 \dots \alpha_n$ (выбираются наименьшие α_j) определяется условие на подмножество переменных прямой задачи, которые в течение итераций данной фазы экспоненциально увеличиваются, пока соответствующий α_j не превысит определенную в начале фазы величину. Внутри итерации после экспоненциального увеличения переменных прямой задачи из выбранного подмножества происходит пересчет левых частей неравенств для $\lambda_1 \dots \lambda_m$ прямой задачи и определяемых ими переменных $y_1, \dots y_n$ двойственной задачи и левых частей неравенств для $\alpha_1 \dots \alpha_n$ двойственной задачи.

Таким образом, алгоритм одновременно ищет решение прямой и двойственной задач. Выполнение алгоритма прерывается, если найдены такие решения прямой и двойственной задач, что либо отношение их значений меньше требуемого мультипликативного фактора $1+\epsilon$, либо по достижении суммой монотонно увеличивающихся переменных двойственной задачи значения 1.

От алгоритма из [7] новый алгоритм отличает простота каждой итерации. В отличие от алгоритма из [4] в новом алгоритме в каждой итерации происходит увеличение не одной, а нескольких переменных прямой задачи. Вследствие этого число итераций нового алгоритма растет медленнее, чем любой полином. Разумный подбор параметров, более динамичный выбор множества увеличиваемых переменных на каждой итерации, гибкие условия окончания работы алгоритма отличают новый алгоритм от алгоритма из [3].

Заметим, что так как приведение задачи к специальной форме приводит к появлению мультипликативной погрешности, равной $(1 - \frac{\epsilon}{\kappa})$, то алгоритм из табл. 1 работает с параметром точности $\epsilon = \frac{\kappa-1}{\kappa} - \frac{\epsilon^2}{\kappa}$, чтобы компенсировать погрешность округления. Выбор $\kappa > 2$, т. е. распределение допустимой погрешности между процедурой округления и самим алгоритмом из табл. 1, предоставляется на усмотрение реализатора. Например, если известно, что матрицы не нуждаются в округлении, то можно положить $\epsilon = \varepsilon$. При реализации алгоритма нами было положено $\kappa = 10$. Кроме этого, при реализации алгоритма параметры ξ и μ можно выбрать так, чтобы выполнялись условия $\xi, \mu > 0$ и $\xi + \mu < \epsilon$.

Выбор конкретных значений параметров ξ, μ и κ может оказать существенное влияние на эффективность реализации. Подчеркнем, что все оценки сложности и ε-оптимальность алгоритма доказаны для произвольных значений параметров ξ, μ и κ, удовлетворяющих упомянутым выше условиям. При реализации алгоритма нами было положено $\xi = 0,45\epsilon$ и $\mu = 0,15\epsilon$.

Докажем оптимальность решения и приведем оценки вычислительной сложности нового алгоритма. Сначала докажем несколько вспомогательных утверждений.

Лемма 2. *В начале алгоритма выполняется неравенство* $Y^0 < 1$.

Справедливость леммы следует из неравенств

$$Y^0 \leqslant me^{\phi\hat{\lambda}_{\max}\frac{\ln m}{\phi\hat{\lambda}_{\max}}-\psi} \leqslant me^{\ln m-\psi} < me^{-\ln m} = 1,$$

где $\hat{\lambda}_{\max} = \max\limits_i \hat{\lambda}_i$ и $\hat{\lambda}_i = \sum\limits_j a_{ij}$.

Лемма 3. *В процессе выполнения алгоритма при любом* j *справедливо неравенство* $\widehat{\alpha} \leqslant \sum\limits_i a_{ij}y_j$, *где* $\widehat{\alpha} = \min\limits_j \alpha_j$.

Доказательство. Так как при выполнении алгоритма двойственные переменные y_i монотонно возрастают, то при любом j

$$\alpha_j \leqslant \sum_i a_{ij}y_i.$$

Таким образом, из определения алгоритма следует, что при любом j

$$\widehat{\alpha} \leqslant \alpha_j \leqslant \sum_i a_{ij}y_i.$$

Лемма 3 доказана.

Лемма 4. *В начале каждой итерации алгоритма при любом* i *выполняется равенство* $\lambda_i = \sum\limits_j a_{ij}x_j$.

Доказательство. Из определения алгоритма (см. табл. 1) следует, что лемма 4 верна для начальной итерации. Пусть лемма 4 верна для итерации $t-1$. Тогда для итерации t имеем

$$\lambda^t = \sum_{j\in Q} a_{ij}\Delta x_j^t + \lambda^{t-1} = \sum_{j\in Q} a_{ij}(x_j^{t-1} + \Delta x_j^t) + \sum_{j\notin Q} a_{ij}x_j^{t-1}$$
$$= \sum_{j\in Q} a_{ij}(x_j^{t-1} + \Delta x_j^t) + \sum_{j\notin Q} a_{ij}x_j^t = \sum_{j\in Q} a_{ij}x_j^t + \sum_{j\notin Q} a_{ij}x_j^t = \sum_j a_{ij}x_j^t.$$

Лемма 4 доказана.

Лемма 5. *В начале каждой итерации алгоритма при любом i выполняется неравенство* $\lambda_i \leqslant 1 - \frac{\xi}{\phi}$.

Доказательство от противного. Пусть существует i такое, что $\lambda_i > 1 - \frac{\xi}{\phi}$. Тогда

$$Y \geqslant y_i \geqslant e^{\phi\lambda_i - \psi} \geqslant e^{\phi\left(1-\frac{\xi}{\phi}\right)-\psi} \geqslant e^{\phi-\xi-\psi} = 1.$$

Противоречие с описанием алгоритма. Лемма 5 доказана.

Лемма 6. *Для любой итерации алгоритма при любом i выполняется неравенство* $\Delta\lambda_i \leqslant \frac{\xi}{\phi}$.

Доказательство. Из определения алгоритма и леммы 4 следует, что на любой итерации t и любом i

$$\lambda_i^t \leqslant \lambda_i^{t-1}\left(1 + \frac{\xi}{\lambda_{\max}^{t-1}\phi}\right) = \lambda_i^{t-1} + \lambda_i^{t-1} \cdot \frac{\xi}{\lambda_{\max}^{t-1}\phi} \leqslant \lambda_i^{t-1} + \frac{\xi}{\phi}.$$

Лемма 6 доказана.

Очевидным следствием этих двух лемм является

Лемма 7. *В процессе работы алгоритма при любом i выполняется неравенство* $\lambda_i \leqslant 1$.

3.1. Оптимальность

Теорема 1. *При любом заданном ε, $0 < \varepsilon \leqslant 1$, алгоритм находит ε-оптимальное решение.*

Доказательство. Из леммы 2 следует, что алгоритм выполнит по крайней мере одну итерацию. Если по окончании итераций выполняется неравенство

$$Y_{\min}\lambda_{\max} < (1+\epsilon)X,$$

то, учитывая, что $Y_{\min} > Y^* = X^*$, где $Y^* = X^*$ — значение оптимума двойственной и прямой задач соответственно, получаем

$$X^*\lambda_{\max} < (1+\epsilon)X.$$

В противном случае, если L — номер последней итерации, то

$$Y^L = \sum_i y_i^L = \sum_i y_i^{L-1} e^{\phi\Delta\lambda_i}.$$

Согласно лемме 6 имеем $\phi\Delta\lambda_i \leqslant \xi \leqslant 1$. Используя неравенство $e^x \leqslant 1 + (1+x)x$, верное для $0 \leqslant x \leqslant 1$, получаем

$$Y^L \leqslant \sum_i y_i^{L-1}(1 + (1+\xi)\phi\Delta\lambda_i) = Y^{L-1} + (1+\xi)\phi\sum_i y_i^{L-1}\sum_{j\in Q}\Delta x_j a_{ij}$$

$$= Y^{L-1} + (1+\xi)\phi\sum_{j\in Q}\Delta x_j\sum_i y_i^{L-1}a_{ij} = Y^{L-1} + (1+\xi)\phi\sum_{j\in Q}\Delta x_j\alpha_j^{L-1}.$$

Согласно лемме 3 при любом $j \in Q$ имеем

$$\alpha_j \leqslant \widehat{\alpha}(1+\mu) \leqslant \min_j \sum_i a_{ij} y_i (1+\mu). \tag{5}$$

Значение допустимого решения двойственной задачи не меньше значения оптимума, т. е. при любом $j \in Q$ из неравенства

$$\frac{Y^L}{\min\limits_j \sum\limits_i a_{ij} y_i^L} \geqslant Y^* = X^*$$

следует неравенство

$$j \in Q \quad \alpha_j^L \leqslant \frac{Y^L}{Y^*}(1+\mu)$$

(напомним, что $X^* = Y^*$ — значение оптимального решения). Поэтому по индукции получаем

$$\begin{aligned} Y^L &\leqslant Y^{L-1}\Big(1 + \frac{\phi(1+\xi)(1+\mu)}{X^*} \sum_{j\in Q} \Delta x_j\Big) \leqslant Y^{L-1} e^{\frac{\phi(1+\xi)(1+\mu)}{X^*} \sum\limits_{j\in Q} \Delta x_j} \\ &\leqslant \Big(\sum_i e^{\phi\lambda_i^0 - \psi}\Big) e^{\frac{\phi(1+\xi)(1+\mu)}{X^*} X} \leqslant \Big(m e^{\phi\hat{\lambda}_{\max} x^0 - \psi}\Big) e^{\frac{\phi(1+\xi)(1+\mu)}{X^*} X} \\ &= \Big(m e^{\phi\hat{\lambda}_{\max}\frac{\ln m}{\phi\hat{\lambda}_{\max}} - \psi}\Big) e^{\frac{\phi(1+\xi)(1+\mu)}{X^*} X} = m e^{\ln m + \frac{\phi(1+\xi)(1+\mu)}{X^*} X - \psi}, \end{aligned}$$

где $X = \sum\limits_j (x_j^L - x^0) = \sum\limits_t \sum\limits_j \Delta x_j^t$ — еще не нормированное на $\lambda_{\max}$ значение найденного алгоритмом решения ($\sum\limits_t$ — суммирование по итерациям). Логарифмируя последнее выражение, получаем

$$\psi - \ln m \leqslant \ln m + \phi(1+\xi)(1+\mu)\frac{X}{X^*}, \quad \frac{X^*}{X} \leqslant \frac{\phi(1+\xi)(1+\mu)}{\psi - 2\ln m}.$$

С другой стороны, согласно лемме 7 по окончании алгоритма имеем $\lambda_{\max} \leqslant 1$. Поэтому

$$\begin{aligned} \frac{X^*\lambda_{\max}}{X} &\leqslant \frac{\phi(1+\xi)(1+\mu)}{\psi - 2\ln m} = \frac{\phi(1+\xi)(1+\mu)}{\phi - (2\ln m + \xi)} \\ &= \frac{(1+\epsilon)(2\ln m + \xi)(1+\xi)(1+\mu)}{(2\ln m + \xi)(1+\epsilon) - (2\ln m + \xi)((1+\epsilon) - (1+\xi)(1+\mu))} \\ &= (1+\epsilon)\frac{(2\ln m + \xi)(1+\xi)(1+\mu)}{(2\ln m + \xi)(1+\xi)(1+\mu)} = 1 + \epsilon = 1 + \frac{\kappa - 1}{\kappa}\varepsilon - \frac{\varepsilon^2}{\kappa}. \end{aligned}$$

Таким образом, учитывая максимальную мультипликативную потерю точности при переходе к специальной форме, получаем, что отношение значения найденного решения задачи в специальной форме, нормированного на $a_{\max}$, к оптимуму исходной задачи в форме (1) не меньше

$$\frac{1-\frac{1}{\kappa}\varepsilon}{1+\frac{\kappa-1}{\kappa}\varepsilon-\frac{\varepsilon^2}{\kappa}}=\frac{\kappa-\varepsilon}{\kappa+(\kappa-1)\varepsilon-\varepsilon^2}=\frac{\kappa-\varepsilon}{(\kappa-\varepsilon)+\varepsilon(\kappa-\varepsilon)}=\frac{1}{1+\varepsilon}.$$

Теорема 1 доказана.

3.2. Оценка максимального числа итераций

Теорема 2. *Общее число итераций алгоритма не превосходит* $O(\varepsilon^{-4}\log^3(\frac{mn}{\varepsilon}))$.

Доказательство. Оценим максимальное число увеличений одного x_j за время выполнения алгоритма. Обозначим это число через I. Согласно лемме 7 имеем

$$\frac{1}{\gamma}x_j\leqslant 1\quad\Rightarrow\quad\frac{\ln m}{\phi n}\Big(1+\frac{\xi}{\phi}\Big)^I\leqslant\gamma\quad\Rightarrow\quad I\leqslant\log_{1+\frac{\xi}{\phi}}\frac{\gamma\phi n}{\ln m}.$$

Таким образом,

$$I\leqslant\log_{1+\frac{\xi}{\phi}}\frac{\gamma\phi n}{\ln m}=O\Big(\frac{\phi}{\xi}\ln\frac{\gamma n\frac{\ln m}{\epsilon-\xi-\mu}}{\ln m}\Big)=O\Big(\frac{\ln m}{\xi(\epsilon-\xi-\mu)}\ln\frac{\gamma n}{\epsilon-\xi-\mu}\Big)$$
$$=O\Big(\frac{\ln m}{\xi(\epsilon-\xi-\mu)}\ln\frac{mn^2}{\epsilon^2(\epsilon-\xi-\mu)}\Big).\quad(6)$$

Теперь оценим число фаз, рассмотрев изменение $\widehat{\alpha}$ при выполнении алгоритма. Из определения алгоритма и леммы 3 следует, что во время работы алгоритма выполняются неравенства

$$\widehat{\alpha}\leqslant\min_j\sum_i a_{ij}y_i\leqslant\sum_i y_i=Y<1,\quad(7)$$

$$\widehat{\alpha}\geqslant\min_j\alpha^0\geqslant\frac{1}{\gamma}e^{-\psi}.\quad(8)$$

С другой стороны, с каждой фазой величина $\widehat{\alpha}$ увеличивается по крайней мере в $1+\mu$ раз. Таким образом, из (7) и (8) получаем

$$\log_{1+\mu}\gamma e^{\psi}=O\Big(\frac{1}{\mu}(\ln\gamma+\psi)\Big)=O\Big(\frac{1}{\mu}\Big(\ln\frac{mn}{\epsilon^2}+\frac{\ln m}{\epsilon-\xi-\mu}\Big)\Big)$$
$$=O\Big(\frac{1}{\mu(\epsilon-\xi-\mu)}\ln\frac{mn}{\epsilon^2}\Big).\quad(9)$$

Из определения алгоритма следует, что на каждой фазе переменная x_q увеличивается в каждой итерации этой фазы. Следовательно, согласно

(6) число итераций в течение одной фазы не превосходит I. Комбинируя (6) и (9), получаем, что общее число итераций не превосходит

$$O\left(\frac{\ln m \cdot \ln \frac{mn}{\epsilon^2} \cdot \ln \frac{mn^2}{\epsilon^2(\epsilon-\xi-\mu)}}{\xi\mu(\epsilon-\xi-\mu)^2}\right)$$

или

$$O(\varepsilon^{-4}\log^3(mn/\varepsilon)). \tag{10}$$

Теорема 2 доказана.

3.3. Оценка числа арифметических операций

Теперь установим верхнюю оценку общего числа арифметических операций, выполненных при работе алгоритма, что определяет верхнюю оценку времени работы последовательной реализации.

Теорема 3. *Вычислительная сложность алгоритма не превосходит* $O(\frac{mn}{\epsilon^2}\ln^2\frac{mn}{\epsilon})$.

Доказательство. Из определения алгоритма следует, что вычислительная сложность (количество арифметических операций) одной итерации равна $O(m|Q|)$. Пусть $|Q^t|$ — число увеличений компонент x на итерации t. Обозначим через Z число увеличений компонент x за время работы алгоритма, т. е.

$$Z = \sum_t |Q^t|.$$

Число фаз обозначим через F. Согласно (9) имеем

$$F = O(\varepsilon^{-2}\ln(mn/\varepsilon)). \tag{11}$$

Вычислительная сложность алгоритма складывается из вычислительной сложности S_F всех выполнений процедуры **Phase** без учета вычислительных затрат процедуры **Iteration**, куда входят выбор $\widehat{\alpha}$ и пересчет α_j после завершения вызовов процедуры **Iteration**, и вычислительных затрат S_I на выполнение процедуры **Iteration**.

Итак, вычислительная сложность S алгоритма равна

$$S_F + S_I = O(mn)\cdot F + O(m)\cdot Z. \tag{12}$$

Осталось оценить Z. Использовав оценку числа изменений j-й компоненты x (обозначим ее Z_j) из (6), получаем

$$Z_j \leqslant O(\varepsilon^{-2}\ln^2(mn/\varepsilon)).$$

Следовательно,

$$Z \leqslant O(\varepsilon^{-2}n\ln^2(mn/\varepsilon)). \tag{13}$$

Из (11)–(13) получаем

$$S \leqslant O(mn) \cdot O(\varepsilon^{-2} \ln(mn/\varepsilon)) + O(m) \cdot O(\varepsilon^{-2} n \ln^2(mn/\varepsilon)).$$

Итак, вычислительная сложность S алгоритма не превосходит

$$O(\varepsilon^{-2} mn \ln^2(mn/\varepsilon)). \tag{14}$$

Теорема 3 доказана.

3.4. Параллельная сложность алгоритма

Приведем оценки параллельной сложности предложенного алгоритма. Будем рассматривать модель EREW PRAM, в которой различным процессорам разрешается запись и чтение только из различных ячеек памяти.

Теорема 4. *При фиксированном ε параллельная сложность алгоритма не превосходит $O((\log mn)^4)$.*

Доказательство. Учитывая полученную оценку числа итераций из (10), остается оценить параллельную сложность одной итерации. В каждой итерации необходимо найти λ_i и α_j при любых i и j, что, по сути, есть задача сложения $O(N)$ чисел (N — число ненулевых элементов матрицы ограничений) $a_{ij}x_j$ и $a_{ij}y_i$ (так как λ_i и α_j являются частичными суммами соответствующих наборов чисел). Понятно, что $O(m)$ независимых операций вычисления экспоненты не изменят полученных оценок. Используя метод сдваивания, параллельную сложность итерации можно оценить сверху величиной

$$O(\log N) = O(\log mn). \tag{15}$$

Учитывая оценки (10), (15), получаем, что при фиксированном ε параллельная сложность алгоритма не превосходит

$$O(\log mn) \cdot O(\varepsilon^{-4} \log^3(mn/\varepsilon)) = O((\log mn)^4).$$

Теорема 4 доказана.

4. Результаты вычислительного эксперимента

Предложенный алгоритм был реализован в последовательной и параллельной версиях. Тестовые задачи ПЛП генерировались программой, которая, получив на вход число переменных, число ограничений и число ненулевых элементов в матрице ограничений, порождала в стандартной форме описание прямой ПЛП-задачи, удовлетворяющей входным данным. Месторасположение и значения ненулевых элементов матрицы

выбирались случайным образом. Точное решение для генерируемых ПЛП-задач находилось с помощью программных продуктов LPSOLVE (симплекс-метод) и PCx (метод внутренней точки). Для сравнения были реализованы некоторые другие алгоритмы приближенного решения задач ПЛП.

Приведем список алгоритмов, используемых в тестировании и сравнении:

- **LPSOLVE** — точное решение задачи ЛП с помощью симплекс-метода (автор M. Berkelaar);
- **PCx** — точное решение задачи ЛП методом внутренней точки (авторы J. Czyzyk, S. Mehrotra, S. J. Wright);
- **GK98** — алгоритм для приближенного решения задачи ПЛП, описанный в [4];
- **BBR97** — алгоритм для приближенного решения задачи ПЛП, описанный в [3];
- **PLPAPXF** — предложенный алгоритм для приближенного решения задачи ПЛП, описанный в разд. 3.3.

Были проведено много вычислительных экспериментов, заключавшихся в решении описанных тестовых задач с различными параметрами точности ε. Некоторые результаты приведены в табл. 2–8.

Для алгоритмов **PLPAPXF**, **BBR97**, **GK98** приведены таблицы, сравнивающие число итераций при нахождении решений для одних и тех же входных данных. Это сделано потому, что число итераций алгоритма зависит не от качества реализации алгоритма, а от его определения. Кроме того, вычислительные сложности каждой итерации алгоритмов **PLPAPXF**, **BBR97**, **GK98** приблизительно одинаковы (перерасчет левых частей ограничений прямой и двойственной задач и соответствующая модификация переменных).

Таблица 2

Число итераций алгоритмов **BBR97**, **GK98**, **PLPAPXF**
в зависимости от параметра требуемой точности ε
для квадратной матрицы порядка 100 *с* 1500 *ненулевыми элементами*

Алгоритм	ε=0.07	ε=0.08	ε=0.09	ε=0.1	ε=0.12	ε=0.14	ε=0.16	ε=0.18	ε=0.2
BBR97	493600	372257	288335	229155	153789	109627	82277	64288	51698
GK98	75441	57382	45039	36238	24824	17985	13574	10569	8433
PLPAPXF	8342	6878	5673	4797	3530	2725	2226	1829	1502

Таблица 3

Число итераций алгоритмов **BBR97**, **GK98**, **PLPAPXF**
в зависимости от параметра требуемой точности ε
для квадратной матрицы порядка 160 *с* 6400 *ненулевыми элементами*

Алгоритм	ε=0.07	ε=0.08	ε=0.09	ε=0.1	ε=0.12	ε=0.14	ε=0.16	ε=0.18	ε=0.2
BBR97	680519	508709	390819	307213	202122	143977	108768	84736	67787
GK98	50963	38757	30416	24469	16757	12137	9157	7128	5686
PLPAPXF	12986	9192	6633	5389	3890	2996	2371	1916	1565

Таблица 4

Число итераций алгоритмов **BBR97**, **GK98**, **PLPAPXF**
в зависимости от параметра требуемой точности ε
для квадратной матрицы порядка 200 *с* 6000 *ненулевыми элементами*

Алгоритм	ε=0.07	ε=0.08	ε=0.09	ε=0.1	ε=0.12	ε=0.14	ε=0.16	ε=0.18	ε=0.2
BBR97	631882	466016	358439	284872	191400	136056	101640	78700	62457
GK98	89634	68173	53504	43045	29483	21357	16117	12548	10011
PLPAPXF	12203	9423	7456	6188	4478	3447	2738	2292	1951

Таблица 5

Число итераций алгоритмов **GK98**, **PLPAPXF**
в зависимости от параметра требуемой точности ε
для квадратной матрицы порядка 300 *с* 22500 *ненулевыми элементами*

Алгоритм	ε=0.07	ε=0.08	ε=0.09	ε=0.1	ε=0.12	ε=0.14	ε=0.16	ε=0.18	ε=0.2
GK98	56416	42900	33665	27079	18541	13426	10128	7882	6286
PLPAPXF	18465	13733	10566	8336	6135	4686	3610	2885	2380

Таблица 6

Число итераций алгоритмов **GK98**, **PLPAPXF**
в зависимости от параметра требуемой точности ε
для квадратной матрицы порядка 500 *с* 62500 *ненулевыми элементами*

Алгоритм	ε=0.07	ε=0.08	ε=0.09	ε=0.1	ε=0.12	ε=0.14	ε=0.16	ε=0.18	ε=0.2
GK98	61685	46902	36800	29598	20260	14667	11061	8606	6861
PLPAPXF	21162	16398	12708	10153	7110	5592	4406	3477	2765

Таблица 7

Число итераций алгоритмов **GK98**, **PLPAPXF**
в зависимости от параметра требуемой точности ε
для квадратной матрицы порядка 1000 *с* 250000 *ненулевыми элементами*

Алгоритм	ε=0.07	ε=0.08	ε=0.09	ε=0.1	ε=0.12	ε=0.14	ε=0.16	ε=0.18	ε=0.2
GK98	67755	51512	40413	32500	22243	16099	12139	9442	7526
PLPAPXF	31198	24378	18623	14862	10309	7611	5854	4771	3905

Таблица 8

Время выполнения (в секундах) алгоритмов точного решения задачи ЛП **LPSOLV**, **PCx** *и алгоритма* **PLPAPXF** *при* $\varepsilon \in \{0.1, 0.2\}$ *в зависимости от размера и числа ненулевых элементов матрицы ограничений с параметрами* <*число строк*> × <*число столбцов*> × <*число ненулевых элементов*>

Алгоритм	100x100x1500	160x160x3840	200x200x6000	700x700x122500	1000x1000 x250000
LPSOLV	2.48	17.47	33.14		7014.07
PCX	1.16	3.53	5.91	326.80	
PLPAPXF $\varepsilon = 0.1$	4.34	10.84	17.48	620.06	1202.80
PLPAPXF $\varepsilon = 0.2$	1.47	3.79	6.35	215.52	513.79

Зависимость числа итераций алгоритмов **BBR97** и **PLPAPXF** от ε — параметра требуемой точности решения приведена в табл. 2–4. Видно значительное преимущество алгоритма **PLPAPXF** над алгоритмом **BBR97**, причем обратная зависимость числа итераций от ε у алгоритма **PLPAPXF** выражена не так сильно, как у алгоритма **BBR97**. Заметим, что алгоритмы **BBR97** и **PLPAPXF** имеют субполиномиальные верхние оценки числа итераций (см. разд. 2 и 3.2).

Зависимость числа итераций алгоритмов **GK98** и **PLPAPXF** от ε — параметра требуемой точности решения приведена в табл. 2–7. Превосходство алгоритма **PLPAPXF** над алгоритмом **GK98** не столь значительно, как над алгоритмом **BBR97**. В отличие от алгоритма **PLPAPXF** алгоритм **GK98** не имеет субполиномиальной верхней оценки числа итераций (см. разд. 2), причем существуют входные данные, на которых число итераций алгоритма **GK98** равно полиномиальной оценке.

В табл. 8 приведена зависимость времени приближенного решения алгоритмом **PLPAPXF** задачи ПЛП с параметрами точности $\varepsilon \in \{0.1, 0.2\}$

и времени точного решения задачи с помощью алгоритмов **LPSOLVE** и **PCx**.

На основании проведенных наблюдений можно сделать следующие выводы.

1. Время работы и число итераций алгоритма **PLPAPXF** существенно меньше времени работы и числа итераций алгоритмов **GK98** и **BBR97**.
2. При $\varepsilon \approx \frac{1}{10}$ выигрыш по времени над алгоритмами точного решения задачи ПЛП начинается с задач, задаваемых квадратными матрицами ограничений порядка 100.

5. Комментарий к статье [3]

Алгоритм приближенного решения задачи ПЛП из [3] приведен в табл. 9. Мы привели его, чтобы анализировать вывод оценки максимального числа итераций данного алгоритма из [3], используя оригинальные обозначения. Здесь мы укажем на то место из [3], которое, по нашему мнению, содержит ошибку. Итак, в разд. 4.3 «Running Time» из [3] в конце доказательства верхней оценки числа итераций в одной фазе приводится следующее равенство:

$$O\Big(\varepsilon^{-1}\phi\ln(\varepsilon^{-1}\gamma mn)+\ln(\gamma m)\Big)=O\Big(\varepsilon^{-1}\ln(\gamma m)\ln(\varepsilon^{-1}\gamma mn)\Big). \qquad (16)$$

Однако из описания алгоритма (см. табл. 9) следует, что

$$\phi=(\rho+\delta)(Q+\rho\ln(Q+\rho\ln(2\rho Q)))\geqslant Q=\Omega(\rho\ln(\gamma m e^{\varepsilon}))=\Omega(r^{-1}\ln(\gamma m)).$$

Следовательно, равенство (6) должно выглядеть следующим образом:

$$O\Big(\varepsilon^{-1}\phi\ln(\varepsilon^{-1}\gamma mn)+\ln(\gamma m)\Big)=O\Big((r\varepsilon)^{-1}\ln(\gamma m)\ln(\varepsilon^{-1}\gamma mn)\Big). \qquad (17)$$

В [3] говорится о том, что число итераций не превосходит

$$O(r^{-1}\varepsilon^{-2}\log^2(\gamma m)\log(\gamma mn/\varepsilon)).$$

При $r\approx\varepsilon$ эта оценка преобразуется в оценку

$$O(\varepsilon^{-3}\log^2(\gamma m)\log(\gamma mn/\varepsilon)),$$

что в $\frac{1}{\varepsilon}$ раз лучше, чем оценка числа итераций из [7], о чем и было упомянуто в [3]. С учетом обнаруженной ошибки при $r\approx\varepsilon$ справедлива следующая оценка числа итераций алгоритма из [3]:

$$O(\varepsilon^{-4}\log^2(\gamma m)\log(\gamma mn/\varepsilon)),$$

что асимптотически эквивалентно оценке числа итераций алгоритма из [7].

Таблица 9

Последовательный алгоритм из [3]

procedure Round-Update()

$\{\forall i \quad \lambda_i = \sum_j a_{ij} y_j;$

$\forall i \quad x_i = \frac{\exp \lambda_i \phi}{\psi};$

$\forall j \quad \alpha_j = \sum_i a_{ij} x_i\};$

procedure Initialize()

$\{$ /* $I_j = \{i \mid a_{ij} > 0\}$ */

/* $I_i = \{i \mid a_{ij} > 0\}$ */

$\delta = (1+\varepsilon)^2; \quad \rho = \frac{1}{r}; \quad Q = \rho \ln(6\gamma m e^{\epsilon});$

$\phi = (r+\delta)(Q + \rho \ln(Q + \rho \ln(2\rho Q)));$

$\psi = m; \quad \psi_F = 6m\frac{\phi}{r+\delta} \exp\left(\frac{\delta\phi}{r+\delta}\right);$

$\forall i \quad \widetilde{n}_i = \sum_{j \in J_i} a_{ij}; \quad \forall j, n_j = \max_{i \in I_j} \widetilde{n}_i; \quad y_j = \frac{\epsilon}{n_j \phi}\};$

Algorithm LP()

{ **call Initialize()**

repeat until $(\psi > \psi_F)$ { /* Phase */

call Round-Update()

repeat until $\left(\min_j \alpha_j \geqslant 1\right)$ { /* Iteration */

$\forall j$, **if** $(\alpha_j < 1)$ **then** $y_j = y_j\left(1 + \frac{\epsilon}{\phi}\right)$

call Round-Update()$\}$

$\psi = \psi(1+\varepsilon)\}$

$\forall j$ **output** $y_j\}$

ЛИТЕРАТУРА

1. **Фомин С. А.** Исследование эффективности реализации нового приближенного алгоритма для задачи положительного линейного программирования // Вопросы кибернетики. Приложения системного программирования. Вып. 4. М.: Науч. совет по комплексной проблеме «Кибернетика», 1998. С. 175–185.
2. **Хачиян Л. Г.** Полиномиальный алгоритм в линейном программировании // Докл. АН СССР. 1979. Т. 244, № 5. С. 1093–1096.

3. **Bartal Y., Byers J. W., Raz D.** Global optimization using local information with application to flow control // 38th annual symp. on foundations of computer science. Proc. Los Alamitos: IEEE Computer Soc. Press, 1997. P. 303–311.
4. **Garg N., Koenemann J.** Faster and simpler algorithms for multicommodity flow and other fractional packing problems // 39th annual symp. on foundations of computer science. Proc. Los Alamitos: IEEE Computer Soc. Press, 1998. P. 300–309.
5. **Grigoriadis M. D., Khachiyan L. G.** Fast approximation schemes for convex programs with many blocks and coupling consfraints // SIAM J. Optimization. 1994. V. 4, N 1. P. 86–107.
6. **Karmarkar N.** A new polynomial-time algorithm for linear programming // Combinatorica. 1984. V. 4, N 4. P. 373–395.
7. **Luby M., Nisan N.** A parallel approximation algorithm for positive linear programming // Proc. of the 25th annual ACM symp. on the theory of computing. New York: ACM Press, 1993. P. 448–457.
8. **Papadimitriou C., Yannakakis M.** Linear programming without the matrix // Proc. of the 25th annual ACM symp. on the theory of computing. New York: ACM Press, 1993. P. 121–129.
9. **Plotkin S., Shmoys D., Tardos E.** Fast approximation algorithms for fractional packing and covering problems // 32th annual symp. on foundations of computer science. Proc. Los Alamitos: IEEE Computer Soc. Press, 1991. P. 495–504.
10. **Trevisan L., Xhafa F.** The parallel complexity of positive linear programming // Parallel Processing Letters. 1998. V. 8, N 4. P. 527–533.

Адрес автора:

Институт системного
программирования,
ул. Б. коммунистическая, 25,
109004 Москва, Россия.
E-mail: fomin@ispras.ru

Статья поступила
13 января 2000 г.