Быстрый приближенный алгоритм для задачи положительного линейного программирования[1]

С. А. Фомин

Аннотация. Предлагается новый алгоритм нахождения ε-оптимального решения задачи положительного линейного программирования, где все исходные данные неотрицательны — $\max\{cx|Ax \le b, x \ge 0\}$. Алгоритм имеет лучшую верхнюю оценку (из известных алгоритмов для этой задачи) вычислительной сложности: $O\left(\frac{N \log \frac{mn}{\varepsilon}}{\varepsilon^2}\right)$, где $m \times n$ — размер матрицы ограничений, а N — количество ненулевых элементов и имеет простую реализацию.

1. Введение

В этой работе будем рассматривать задачи положительного линейного программирования (ПЛП), также называемые задачами дробной упаковки, когда требуется найти вектор $\mathbf{x} = (x_1, \ldots, x_n)$, такой что:

$$\mathbf{cx} \to max, \quad \mathcal{A}\mathbf{x} \le \mathbf{b}, \quad x_j \ge 0,$$

где $\mathcal{A}$ — рациональная $m \times n$ матрица, $a_{ij} \ge 0$, $\mathbf{c}$ — рациональный вектор, $c_j \ge 0$, $\mathbf{b}$ — рациональный вектор, $b_i \ge 0$, и задачи дробного покрытия (когда линейные ограничения есть ограничения вида $\mathcal{A}\mathbf{x} \ge \mathbf{b}$, 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 [1, 2]. Несмотря на существование полиномиальных алгоритмов решения задач линейного программирования [4, 5], рост размерности задачи, коммуникационные аспекты, возможность параллельной реализации подталкивают к поиску приближенных алгоритмов для решения задач ПЛП. В последние годы было разработано много ε–приближенных ПЛП-алгоритмов [6, 7, 8, 1, 2, 9, 10].

В данной работе мы предлагаем новый алгоритм для решения задачи ПЛП, который имеет лучшие оценки вычислительной сложности и простую программную реализацию. Для предложенного алгоритма доказана

[1]Работа выполнена при поддержке РФФИ, проект 02–01–00713.

оценка вычислительной сложности $O\left(\frac{N \log \frac{mn}{\varepsilon}}{\varepsilon^2}\right)$, где N — число ненулевых элементов в матрице ограничений, что лучше оценок для предшествующих алгоритмов.

2. Постановка задачи ПЛП

Рассмотрим постановки задач ПЛП:

Задача 1. ***ПЛП типа упаковки***

Имеется неотрицательная матрица ограничений $\mathcal{A}$: m строк, n столбцов, $a_{ij} \ge 0$, положительные вектора $\mathbf{c} = \langle c_1, \ldots, c_n \rangle$, $c_j > 0$ и $\mathbf{b} = \langle b_1, \ldots, b_m \rangle$, $b_i > 0$.

$$\begin{aligned} & \sum_{j=1}^{n} c_j x_j \to \max \\ \forall i \quad & \sum_{j=1}^{n} a_{ij} x_j \le b_i \\ \forall j \quad & x_j \ge 0. \end{aligned}$$

Задача 2. ***ПЛП типа покрытия***

Имеется неотрицательная матрица ограничений $\mathcal{A}$: m строк, n столбцов, $a_{ij} \ge 0$, положительные вектора $\mathbf{c} = \langle c_1, \ldots, c_n \rangle$, $c_j > 0$ и $\mathbf{b} = \langle b_1, \ldots, b_m \rangle$, $b_i > 0$.

$$\begin{aligned} & \sum_{j=1}^{n} c_j x_j \to \min \\ \forall i \quad & \sum_{j=1}^{n} a_{ij} x_j \ge b_i \\ \forall j \quad & x_j \ge 0. \end{aligned}$$

Далее, везде n — это число переменных, m — число ограничений прямой задачи (не считая ограничений вида $x_i \ge 0$), а N — число ненулевых элементов матрицы ограничений. Мы будем использовать j для индексирования переменных прямой задачи и столбцов матрицы ограничений, и, где не будет явно указано обратное, будем считать, что j изменяется в диапазоне $1 \ldots n$. Также мы будем использовать i для индексирования ограничений, и, соответственно, по умолчанию считаем, что $i \in [1 \ldots m]$.

Заметим, что задачи 1 и 2 можно представить в так называемой *стандартной форме* (см. [1]), где векторы стоимостей $\mathbf{c}$ и правых частей ограничений $\mathbf{b}$ состоят из единиц. Эта форма интересна тем, что одна матрица

ограничений $\mathcal{A}$ используется как в прямой задаче упаковке, так и в двойственной ей задаче покрытия:

Задача 3. ***Прямая задача ПЛП***

Дана неотрицательная матрица ограничений $\mathcal{A}$: m строк, n столбцов, $a_{ij} \geq 0$.

$$X \equiv \sum_j x_j \to \max$$
$$\forall i \qquad \lambda_i \equiv \sum_j a_{ij} x_j \leq 1$$
$$\forall j \qquad x_j \geq 0.$$

Задача 4. ***Двойственная задача ПЛП***

Дана неотрицательная матрица ограничений $\mathcal{A}$: m строк, n столбцов, $a_{ij} \geq 0$.

$$Y \equiv \sum_i y_i \to \min$$
$$\forall j \qquad \alpha_j \equiv \sum_i a_{ij} y_i \geq 1$$
$$\forall i \qquad y_i \geq 0.$$

Сразу заметим, что ПЛП-задача 1, легко преобразуется в ПЛП-задачу 3, посредством преобразования матрицы ограничений:

$$\hat{a}_{ij} = \frac{a_{ij}}{b_i c_j}. \tag{1}$$

Очевидно такое преобразование не повлияет на значение целевой функции. А решения $\hat{\mathbf{x}}^*$ задачи 3 и $\mathbf{x}^*$ задачи 1 соотносятся очевидным образом:

$$x_j^* = \frac{\hat{x}^*}{c_j}.$$

Ниже (см. параграф 2.1) мы покажем, что без потери общности мы можем считать, что для коэффициентов матрицы ограничений входной задачи ПЛП выполняется $a_{min} \leq a_{ij} \leq a_{max}$, где $\frac{a_{max}}{a_{min}} = O((\frac{l}{\varepsilon})^{const})$, $l = \min(m, n)$.

В случае, если имеется экспоненциальный разброс в коэффициентах, то можно провести округление, в результате которого, для данного τ, $\frac{a_{max}}{a_{min}} = O(\frac{l^2}{\tau^2})$, а значение решение округленной задачи 3 будет не меньше $(1 - \tau)$, умноженного на значение решения неокругленной задачи.

2.1. Округление

Обозначим $a_j^* = \max_i a_{ij}$, $a^* = \min_j a_j^*$, $l = \min(m, n)$.

Пусть $a_{max} = \frac{2la^*}{\tau}$. Тогда для всех $a_j^* > a_{max}$, зафиксируем $x_j \equiv 0$. Заметим, что значение оптимального решения исходной задачи $X^* \geq \frac{1}{a^*}$, что следует из допустимости решения $x_k = \frac{1}{a^*}$, где $k = \operatorname{argmin}_j a_j^*$.

Для потерь из-за округления ΔX с одной стороны $\Delta X \leq \frac{n}{a_{max}}$, с другой $\Delta X \leq \frac{m}{a_{max}}$. Получаем:

$$\Delta X \leq \frac{l}{a_{max}} \leq \frac{\tau}{2a^*} \leq \frac{\tau}{2} X^*. \tag{2}$$

Теперь, пусть $a_{min} = \frac{\tau a^*}{2l}$. Для всех $a_{ij} < a_{min}$, положим $a_{ij} \equiv a_{min}$.

Заметим также, что $X^* = Y^* \leq \frac{l}{a^*}$, что следует из допустимости двойственного решения $\forall i \in S$ $y_i = \frac{1}{a^*}$, где S-минимальное множество ограничений, в котором участвуют все переменные: $|S| \leq m,\ |S| \leq n \to |S| \leq l$.

Оценим максимальное увеличение левых частей округленных ограничений, при подстановке оптимального решения неокругленной задачи 3:

$$\Delta\lambda_{max} \leq X^* \cdot a_{min} \leq \frac{\tau a^*}{2l} \frac{l}{a^*} \leq \frac{\tau}{2}.$$

Отсюда для значения округленного оптимума будет

$$X' \geq \frac{X^*}{1 + \frac{\tau}{2}}. \tag{3}$$

Из (2) и (3) имеем, что задача, с коэффициентами, усеченными по заданным a_{max} и a_{min}, имеет решение допустимое для исходной задачи 3, и для его значения выполняется

$$X' \geq \frac{1 - \tau/2}{1 + \tau/2} X^* \geq (1 - \tau) X^*.$$

3. Обзор алгоритмов для приближенного решения задачи ПЛП

Определение 1. *Приближенный алгоритм является* ***алгоритмом с мультипликативной точностью*** *D, если он при любых исходных данных находит допустимое решение со значением целевой функции, отличающемся от оптимума не более, чем в D раз.*

Определение 2. *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, отличающемся от оптимума не более, чем в $(1+\varepsilon)$ раз.

В [7] был описан ε–оптимальный алгоритм для сопряженных задач 1 и 2 состоящий из $O(\frac{mn/\varepsilon}{\varepsilon^4})$ итераций сложности $O(mn)$.

Затем в [1] был описан приближенный алгоритм который, что для заданных $\varepsilon > 0$ и $0 < r < \ln(\frac{m^3}{\varepsilon^2})$ находил допустимые решения $\mathbf{x}$ и $\mathbf{y}$, для сопряженных задач 3 и 4 соответственно, что $\sum_j x_j \geq \frac{\sum_i y_i}{r+(1+\varepsilon)^2}$. При $r \approx \varepsilon$ алгоритм можно было рассматривать, как ε–оптимальный.

Алгоритм также состоял из итераций сложности $O(mn)$, причем в статье [1] число итераций оценено, как $O(\frac{\log^2(\gamma m)\log(\gamma mn/\varepsilon)}{r\varepsilon^2})$, где $\gamma = \frac{m^2}{\varepsilon^2}$, что при $r \approx \varepsilon$ эквивалентно $O(\frac{\log^3(\frac{mn}{\varepsilon})}{\varepsilon^3})$. К сожалению, в анализе времени выполнения, приведенном в [1] нами обнаружена ошибка, и согласно нашим оценкам, правильная оценка числа итераций будет $O(\frac{\log(\gamma m)^2\log(\gamma mn/\varepsilon)}{r^2\varepsilon^2})$ или (при $r \approx \varepsilon$) $O(\frac{\log^3(\frac{mn}{\varepsilon})}{\varepsilon^4})$, что асимптотически эквивалентно числу итераций алгоритма из [7].

В работе [2] был представлен ε–оптимальный алгоритм решения задачи ПЛП, для которого доказана верхняя оценка числа итераций $O(\frac{m}{\varepsilon^2}\log(m))$, каждая итерация сложности $O(N)$, (напомним, что N обозначает число ненулевых элементов в матрице ограничений задачи ПЛП). Заметим, что при фиксированном ε алгоритмы из работ [7, 1] имеют полилогарифмические верхние оценки числа итераций, а алгоритм из [2] имеет полиномиальную верхнюю оценку, причем существуют входные данные, на которых эта оценка достигается.

В работе [9] был представлен ε–оптимальный алгоритм для сопряженных задач 1 и 2 временной сложности $O(mn\frac{\log^2(mn/\varepsilon)}{\varepsilon^2})$.

В статье [10] рассмотрен более общий класс задач — приближенное решение *смешанной* задачи типа покрытия и упаковки[2]:

Задача 5. ***Смешанная ПЛП задача типа покрытия и упаковки:***

Даны неотрицательные матрицы $\boldsymbol{P}$,$\boldsymbol{C}$, вектора $\boldsymbol{p}$,$\boldsymbol{c}$, $\epsilon \in (0,1)$, необходимо найти ***допустимый*** *вектор $\boldsymbol{x} \geq 0$, для которого*

$$\begin{aligned} \boldsymbol{Px} &\leq (1+\epsilon)\boldsymbol{p} \\ \boldsymbol{Cx} &\geq \boldsymbol{c}. \end{aligned}$$

В [10] представлен алгоритм для решения задачи 5 с вычислительной

[2]Approximate Mixed Packing and Covering

сложностью $O(\frac{m^2\log(m)}{\epsilon^2})$. Однако, хотя теоретически задача 1 сводится к задаче 5, представленный в [10] алгоритм сведения достаточно сложен.

В следующем параграфе, мы представим алгоритм для решения непосредственно оптимизационных задач 1 и 2, обладающий аналогичной асимптотической оценкой сложности — $O\left(\frac{N\log\frac{mn}{\varepsilon}}{\varepsilon^2}\right)$.

x_j	Переменные задачи 3
y_i	Переменные задачи 4
X	Значение целевой функции задачи 3
Y	Значение целевой функции задачи 4
$\hat{Y}$	Верхняя граница для Y
$\tilde{Y}$	Верхняя оценка для значения оптимума задач 3 и 4
Δx_j	Дискретные приросты переменных задачи 3
λ_i	Значение левых частей ограничений задачи 3
λ_{max}	Максимальное значение λ_i. $\frac{\mathbf{x}}{\lambda_{max}}$ — допустимое решение задачи 3.
α_j	Значение левых частей ограничений задачи 4
α_{min}	Минимальное значение α_j. $\frac{\mathbf{y}}{\alpha_{min}}$ — допустимое решение задачи 4.
ξ, μ	Параметры сходимости. Необходимое условие: $\xi + \mu + \xi\mu < \epsilon$.

Вход: m,n, $\epsilon \in (0,1)$ неотрицательная $m \times n$ матрица $\mathcal{A}$.
Выход: x, y — ϵ-оптимальные решения задач 3 и 4.

$\xi = \mu \leftarrow \frac{\epsilon}{3}$, $\quad \chi = (1+\xi)(1+\mu)$
$\hat{Y} \equiv e^{\frac{(1+\epsilon)\log m + \xi\chi}{1+\epsilon-\chi}}$
$\forall j \;\; \Delta x_j \leftarrow \frac{\xi}{a_j^*} \equiv \frac{\xi}{\max_i a_{ij}}$
$\forall j \;\; x_j \leftarrow 0, \quad X \leftarrow 0 \quad \{X \equiv \sum_j x_j)\}$
$\forall i \;\; y_i \leftarrow 1, \quad Y \leftarrow m \quad \{\forall i \;\; y_i \equiv e^{\lambda_i}, Y \equiv \sum_i y_i)\}$
$\forall j \;\; \alpha_j \leftarrow \sum_i a_{ij}y_i, \quad \alpha_{min} \leftarrow \min_j \alpha_j$
repeat
  for all j : $\frac{Y}{\alpha_j} \geq \frac{\tilde{Y}}{(1+\mu)}$ **do**
    $x_j \leftarrow x_j + \Delta x_j$
    $\forall i \; \lambda_i \leftarrow \sum_j a_{ij}x_j, \quad \lambda_{max} \leftarrow \max_i \lambda_i \;, \quad X \leftarrow \sum_j x_j$
    $\forall i \;\; y_i \leftarrow e^{\lambda_i}, \quad Y \leftarrow \sum_i y_i$
    $\forall j \;\; \alpha_j \leftarrow \sum_i a_{ij}y_i, \;\; \alpha_{min} \leftarrow \min_j \alpha_j$
    $\tilde{Y} \leftarrow \frac{Y}{\alpha_{min}}$
  end for
until $(\frac{X}{\lambda_{max}} \geq \frac{\tilde{Y}}{(1+\epsilon)}$ or $Y > \hat{Y})$
return $\frac{\mathbf{x}}{\lambda_{max}}, \frac{\mathbf{y}}{\alpha_{min}}$.

Алгоритм 1: Упрощенная идея алгоритма PLPAPX

4. Новый алгоритм для приближенного решения задачи ПЛП

Поясним основные идеи предлагаемого алгоритма (см. алгоритмы 1,2). Алгоритм является усовершенствованием алгоритма из [9], но его идея алгоритма близка к идеям алгоритмов из [7, 1, 2], содержащих разные формы применения релаксаций Лагранжа, и основана на последовательном увеличении переменных прямой задачи-упаковки в зависимости от переменных двойственной задачи-покрытия, соответствующих ограничениям прямой задачи, т.е. алгоритм одновременно ищет решение прямой и двойственной задачи.

Сначала поясним основную идею алгоритма (см. алгоритм 1). Получив на вход матрицу ограничений ПЛП-задачи, алгоритм инициализирует параметры (μ, ξ, Δx_j), определяющие его поведение, и обнуляет значения переменных задачи-упаковки x.

Далее, основная работа алгоритма состоит из последовательного увеличения переменных x, и перерасчета вектор-переменных λ, y, α (в указанном порядке). На основе значений α вновь выбираются переменные x_j для увеличения, и цикл повторяется.

Выполнение алгоритма прерывает достижение оптимальности. Действительно, если найдены решения прямой и двойственной задачи, отношение значений которых меньше требуемого мультипликативного фактора $(1+\epsilon)$, то, т.к. оптимумы прямой и двойственной задачи совпадают, имеем

$$\frac{X}{\lambda_{max}} \geq \frac{\tilde{Y}}{(1+\epsilon)} \geq \frac{1}{1+\epsilon} Y^* = \frac{1}{1+\epsilon} X^*.$$

Либо, в случае если $Y > \hat{Y}$, оптимальность гарантируется леммой 2.

Теперь рассмотрим алгоритм 2. Основная проблема алгоритма 1, это неоптимальный перерасчет зависящих от $\mathbf{x}$ переменных, при увеличении каждого компонента x_j. Зависимые переменные можно подразделить на две группы. В первую войдут переменные λ, λ_{max}, y, X, Y, которых можно перерасчитать при увеличении компоненты x_j, за время $O(|I_j|)$. Тем самым, вклад сложности этих операций в общую вычислительную сложность алгоритма можно оценить как $O(|I_j|)$ на число увеличений x_j (см. лемму 4).

Переменные же α зависят от $\mathbf{x}$ сложным образом, и необходимо обязательно пересчитать α_j перед принятием решения, увеличивать соответствующий x_j, или нет.

Поэтому, глобальный пересчет α, имеющий сложность в худшем случае

$O(N)$, происходит не более двух раз внутри внешнего цикла алгоритма, и его вклад в сложность алгоритма можно оценить как $O(N)$ на число внешних циклов (см. лемму 3).

Подобное разбиение позволяет улучшить окончательную оценку вычислительной сложности (см. теорему 1). Использование $\hat{\alpha}_{min}$ для частичного пересчета α, не позволяет улучшить оценку вычислительной сложности, но существенно улучшают эффективность реализации.

Напомним, что если проводилось округление с параметром τ (см. параграф 2.1) то могла появится мультипликативная погрешность не превышающая $(1-\tau)$ (см. параграф 2.1). Поэтому, если входной параметр точности ε, параметр округления τ, то алгоритм 2 следует запускать с параметром точности $\epsilon = \varepsilon - \tau - \varepsilon\tau$, чтобы компенсировать погрешность округления. Тогда мы получим, что мультипликативная точность, с учетом округления:

$$\frac{1-\tau}{1+\epsilon} = \frac{1-\tau}{(1-\tau)(1+\varepsilon)} = \frac{1}{1+\varepsilon}.$$

Выбор $\tau < \varepsilon$, т.е. распределение допустимой погрешности между процедурой округления и самим алгоритмом 2 предоставляется на усмотрение реализатора. Например, если известно, что матрицы не нуждаются в округлении, то можно положить $\epsilon = \varepsilon$.

Кроме этого при реализации можно выбрать параметры ξ и μ, при условии, что $\xi, \mu > 0$ и $\xi + \mu + \xi\mu < \epsilon$. И все же, хотя выбор конкретных значений параметров ξ, μ и τ может оказать существенное влияние на эффективность реализации, все оценки сложности и ε-оптимальность алгоритма доказаны для произвольных значений параметров ξ, μ и τ, удовлетворяющих вышеупомянутым условиям.

Ниже мы приведем оценки вычислительной сложности нового алгоритма и доказательство оптимальности.

Сначала небольшая вспомогательная лемма. Из определения алгоритма 2 непосредственно следует:

Лемма 1. *При увеличении любого x_j: $\forall i \;\; \Delta\lambda_i \leq \xi$ и $\exists i \; \Delta\lambda_i = \xi$.*

Теперь оптимальность:

Лемма 2. *Если при выполнении алгоритма выполнилось условие $Y > \hat{Y}$, то найдено ϵ-оптимальное решение.*

Доказательство. Выведем оценку зависимости Y от X. Назовем *шагом* увеличение компоненты x в алгоритме. Пусть L номер некоторого шага,

I_j	$\{i : a_{ij} > 0\}$ индексы строк с ненулевым коэффициентом для столбца j
α_{min}	Нижняя оценка для $\min_j \alpha_j$. Нужна для выбора увеличиваемых x_j.
$\hat{\alpha}_{min}$	Верхняя оценка для $\min_j \alpha_j$. Нужна для выбора пересчитываемых α_j.

Вход: m,n, $\epsilon \in (0,1)$ неотрицательная $m \times n$ матрица $\mathcal{A}$.
Выход: $\mathbf{x}$, $\mathbf{y}$ — ϵ-оптимальные решения задач 3 и 4 соответственно.

$\xi = \mu \leftarrow \frac{\epsilon}{3}$, $\quad \chi = (1+\xi)(1+\mu)$
$\hat{Y} \equiv e^{\frac{(1+\epsilon)\log m + \xi\chi}{1+\epsilon-\chi}}$
$\forall j \;\; \Delta x_j \leftarrow \frac{\xi}{a^*_j} \equiv \frac{\xi}{\max_i a_{ij}}$
$\forall j \;\; x_j \leftarrow 0$, $\quad X \leftarrow 0$
$\forall i \;\; \lambda_i \leftarrow 0$, $\quad \lambda_{max} \leftarrow \max_i \lambda_i$
$\forall i \;\; y_i \leftarrow 1$, $\quad Y \leftarrow m$
$\forall j \;\; \alpha_j \leftarrow \sum_i a_{ij}$, $\quad \alpha_{min} \leftarrow \min_j \alpha_j$
repeat
  for all $j : \frac{Y}{\alpha_j} \geq \frac{\tilde{Y}}{(1+\mu)}$ **do**
    $\alpha_j \leftarrow \sum_i a_{ij} y_i$
    while $\frac{Y}{\alpha_j} \geq \frac{\tilde{Y}}{(1+\mu)}$ and $Y < \hat{Y}$ **do**
      $x_j \leftarrow x_j + \Delta x_j$, $\quad X \leftarrow X + \Delta x_j$ $\quad$ {Прирост x_j}
      for all $i \in I_j$ **do**
        $\Delta\lambda_i \leftarrow a_{ij}\Delta x_j$, $\quad \lambda_i \leftarrow \lambda_i + \Delta\lambda_i$, $\quad \lambda_{max} \leftarrow \max(\lambda_{max}, \lambda_i)$
        $\Delta y_i \leftarrow y_i(e^{\Delta\lambda_i} - 1)$, $\quad y_i \leftarrow y_i + \Delta y_i$, $\quad Y \leftarrow Y + \Delta y_i$
        $\alpha_j \leftarrow \alpha_j + a_{ij}\Delta y_i$
      end for
      $\hat{\alpha}_{min} = \alpha_j$.
    end while
  end for
  $\forall j \in \{j : \alpha_j \leq \hat{\alpha}_{min}(1+\mu)\}$ $\alpha_j \leftarrow \sum_i a_{ij} y_i$, $\quad \alpha_{min} \leftarrow \min_j \alpha_j$, $\quad \tilde{Y} = \frac{Y}{\alpha_{min}}$
until $(\frac{Y\lambda_{max}}{X\alpha_{min}} \leq (1+\epsilon)$ or $Y \geq \hat{Y})$
return $\frac{\mathbf{x}}{\lambda_{max}}, \frac{\mathbf{y}}{\alpha_{min}}$.

Алгоритм 2: Алгоритм PLPAPX

а j— индекс увеличившейся на этом шаге компоненты x.

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

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

$$\begin{aligned}
\hat{Y} \leq Y^L &\leq \sum_i y_i^{L-1}(1 + (1+\xi)\Delta\lambda_i) \\
&= Y^{L-1} + (1+\xi)\sum_i y_i^{L-1}\Delta x_j a_{ij} \\
&= Y^{L-1} + (1+\xi)\Delta x_j \sum_i y_i^{L-1} a_{ij} \\
&= Y^{L-1} + (1+\xi)\Delta x_j \alpha_j^{L-1}.
\end{aligned}$$

С другой стороны, для любого шага L и выбранного на нем индекса j должно выполнятся

$$\frac{Y^{L-1}}{\alpha_j^{L-1}} < \frac{\tilde{Y}}{1+\mu} < \frac{Y^*}{1+\mu} \equiv \frac{X^*}{1+\mu}, \tag{4}$$

(напомним, что $X^* = Y^*$ — значение оптимального решения), или

$$\alpha_j^{L-1} > \frac{Y^{L-1}(1+\mu)}{X^*}. \tag{5}$$

Получаем по индукции:

$$\begin{aligned}
Y^L &\leq Y^{L-1}\left(1 + \frac{(1+\xi)(1+\mu)}{X^*}\Delta x_j\right) \\
&\leq Y^{L-1} e^{\frac{(1+\xi)(1+\mu)}{X^*}\Delta x_j} \\
&\leq Y^0 e^{\frac{(1+\xi)(1+\mu)}{X^*}X} \\
&= m e^{\frac{(1+\xi)(1+\mu)}{X^*}X}.
\end{aligned}$$

Логарифмируя последнее выражение, получаем:

$$\frac{X^*}{X} \leq \frac{(1+\xi)(1+\mu)}{\log(Y^L/m)} = \frac{\chi}{\log(Y^L/m)}.$$

Оценим λ_{max}^L. Во-первых, заметим, что на шаге $L-1$:

$$\lambda_{max}^{L-1} \leq \log \sum_i e^{\lambda_i^{L-1}} = \log Y^{L-1} \leq \log \hat{Y}.$$

Во-вторых, учитывая лемму 1, имеем

$$\lambda_{max}^L \leq \lambda_{max}^{L-1} + \xi \leq \log \hat{Y} + \xi.$$

Итак, оценим отношение оптимума $X*$ к значению решения $\frac{X}{\lambda_{max}^L}$:

$$\begin{aligned}
\frac{X^*\lambda_{max}^L}{X} &\leq \frac{X^*(\log\hat{Y}+\xi)}{X} \\
&\leq (\log\hat{Y}+\xi)\frac{(1+\xi)(1+\mu)}{\log(Y^L/m)} \\
&\leq (\log\hat{Y}+\xi)\frac{\chi}{\log(\hat{Y})-\log m} \\
&= \frac{(1+\epsilon)\log m+\xi\chi+\xi(1+\epsilon-\chi)}{(1+\epsilon-\chi)}\frac{\chi}{\frac{(1+\epsilon)\log m+\xi\chi-(1+\epsilon-\chi)\log m}{(1+\epsilon-\chi)}} \\
&= \frac{\chi(1+\epsilon)(\log m+\xi)}{\chi(\log m+\xi)} = (1+\epsilon).
\end{aligned}$$

□

4.1. Оценка вычислительной сложности

Сначала оценим вычислительную сложность пересчета α_j, обозначим ее C_α.

Лемма 3. $C_\alpha = O(\frac{N\log\frac{mn}{\varepsilon}}{\varepsilon^2})$

Доказательство. Оценим число внешних циклов. Очевидно, что значение α_{min}, с каждой итерацией внешнего цикла должно увеличиваться не менее чем в $1+\mu$ раз. Учитывая, что $\alpha_{min}^L \leq a_{max}Y^L$ и $\alpha_{min}^0 \geq a_{min}$, а верхняя оценки сложности пересчета — $O(N)$, получаем

$$\begin{aligned}
C_\alpha &= O\left(N\log_{1+\mu}\frac{\alpha_{min}^L}{\alpha_{min}^0}\right) = O\left(N\frac{1}{\mu}\log\frac{\alpha_{min}^L}{\alpha_{min}^0}\right) \\
&= O\left(N\frac{1}{\varepsilon}\log\frac{a_{max}e^{\frac{\log m}{\varepsilon}}}{a_{min}}\right) = O\left(N\frac{\log\frac{mn}{\varepsilon}}{\varepsilon^2}\right).
\end{aligned}$$

□

Теперь оценим вычислительную сложность остальных операций, обозначим ее C_x.

Лемма 4.

$$C_x = O\left(N\frac{\log\frac{mn}{\varepsilon}}{\varepsilon^2}\right).$$

Доказательство. Сначала оценим максимальное число увеличений каждой из переменных x_j. Заметим, что для каждого j существует по крайней мере один λ_i, который будет увеличиваться на ξ, при каждом увеличении x_j. При этом сложность пересчета переменных после увеличения x_j будет $O(I_j)$. Таким образом,

$$\begin{aligned}
C_x &= \sum_j O(I_j)\frac{\lambda_{max}^L}{\xi} = O\left(\sum_j I_j\frac{\log\hat{Y}}{\varepsilon}\right) = O\left(\sum_j I_j\frac{\log\frac{mn}{\varepsilon}}{\varepsilon^2}\right) \\
&= O\left(N\frac{\log\frac{mn}{\varepsilon}}{\varepsilon^2}\right).
\end{aligned}$$

□

Из лемм 3 и 4 получаем оценку общей сложности алгоритма.

Теорема 1. *Вычислительная сложность алгоритма равна* $O\left(N\frac{\log\frac{mn}{\varepsilon}}{\varepsilon^2}\right)$.

5. Вычислительные эксперименты

Ниже мы приведем результаты ряда вычислительных экспериментов (см. таблицу 1).

Тестовыми данными были задачи линейного программирования, где векторы стоимости и ограничений были заполнены единицами: $\mathbf{c} = (1,\ldots,1)$, $\mathbf{b} = (1,\ldots,1)$, а 0/1-матрица ограничений порождалась случайным размещением заданного количества ненулевых элементов (единиц).

Время решения тестовых задач (в секундах) приводится в сравнении с временем точного решения этих задач алгоритмами симплекс-метода и метода внутренней точки, реализованных в пакете GLPK [11].

По результатам [12] регулярного тестирования различных пакетов линейной оптимизации, пакет GLPK является лидером среди некоммерческих пакетов, и несильно уступает ведущим коммерческим программам для линейной оптимизации.

Проведенное тестирование показало высокую эффективность и обоснованность использования алгоритма PLPAPX для приближенного решения задач ПЛП.

Информация о процессоре используемом в тестировании:

```
Model : AMD Athlon(tm) XP 2000+
Co-Processor (FPU) : Built-in
```

Строк	Столбцов	Ненулевых	Симплекс метод	Метод ВТ	PLPAPX $\varepsilon = 0.1$
1000	1000	00200000	207.153	418.235	1.297
1000	1000	00300000	218.901	495.268	1.297
1000	1000	00400000	311.504	591.932	0.953
1000	1000	00500000	336.478	662.848	0.844
1000	4000	00400000	432.328	1097.92	19.508
1000	4000	00800000	896.733	1820.3	14.303
1000	4000	01200000	1380.57	2515.66	14.752
1000	4000	01600000	1962.68	3068.66	16.438
1000	4000	02000000	2060.53	3870.04	16.203
1000	7000	00700000	599.666	1961.27	38.157
1000	7000	01400000	1176.04	4070.44	37.594
1000	7000	02100000	1827.29	6173.1	52.401
1000	7000	02800000	2063.51	7697.57	60.343
1000	7000	03500000	3157.05	10450	58.719
4000	1000	00400000	869.033	*	6.677

Таблица 1: Результаты вычислительных экспериментов

```
Speed : 1.67GHz
Performance Rating : PR2427 (estimated)
```

Литература

[1] Yair Bartal, John W. Byers, and Danny Raz. Global optimization using local information with application to flow control. In *IEEE FOCS*, pages 303–311, 1997.

[2] N. Garg and J. Koenemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. In *IEEE FOCS*, pages 300–309, 1998.

[3] Luca Trevisan and Fatos Xhafa. The parallel complexity of positive linear programming. *Parallel Processing Letters*, 8(4):527–533, 1998.

[4] L.G. Khachiyan. A polynomial-time algorithm for linear programming. *Soviet Math. Dokl.*, 20(1):191–194, 1979.

[5] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–396, 1984.

[6] S. Plotkin, D. Shmoys, and E. Tardos. Fast approximation algorithms for fractional packing and covering problems. In *Proc. 32nd IEEE FOCS 91*, pages 495–504, 1991.

[7] M. Luby and N. Nisan. A parallel approximation algorithm for positive linear programming. In *Proc. of 25th ACM STOC*, pages 448–457, 1993.

[8] M.D.Grigoriadis and L.G.Khachiyan. Fast approximation schemes for convex programs with many blocks and coupling constraints. *SIAM J. Optimization*, 4:86–107, 1994.

[9] С.А. Фомин. Новый приближенный алгоритм для решения задачи положительного линейного программирования. *Дискретный анализ и исследование операций. Серия 2.*, 8(2), 2001.

[10] Neal E. Young. Sequential and parallel algorithms for mixed packing and covering. In *IEEE Symposium on Foundations of Computer Science*, pages 538–546, 2001.

[11] Andrew Makhorin. GLPK Reference Manual. `http://www.gnu.org/software/glpk/glpk.html`, 2003.

[12] Hans Mittelmann. Benchmarks for Optimization Software. Department of Mathematics and Statistics, Arizona State University, `http://plato.la.asu.edu/bench.html, 2003.`