УДК 519.852 **DOI: 10.14529/cmse230201**

О НОВОЙ ВЕРСИИ АПЕКС-МЕТОДА ДЛЯ РЕШЕНИЯ ЗАДАЧ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ*

© 2023 Л.Б. Соколинский, И.М. Соколинская

Южно-Уральский государственный университет
(454080 Челябинск, пр. им. В.И. Ленина, д. 76)
E-mail: leonid.sokolinsky@susu.ru, irina.sokolinskaya@susu.ru

Поступила в редакцию: 07.04.2023

В статье представлена новая версия масштабируемого итерационного метода линейного программирования, получившего название «апекс-метод». Ключевой особенностью этого метода является построение пути, близкого к оптимальному, на поверхности допустимой области от определенной начальной точки до точного решения задачи линейного программирования. Оптимальный путь — это путь движения по поверхности многогранника в направлении максимального увеличения или уменьшения значения целевой функции в зависимости от того, ее максимум или минимум необходимо найти. Апекс-метод основан на схеме предиктор-корректор и состоит из двух стадий: Quest (предиктор) и Target (корректор). На стадии Quest вычисляется грубое начальное приближение задачи линейного программирования. Основываясь на этом начальном приближении, на стадии Target вычисляется решение задачи линейного программирования с заданной точностью. Основная операция, используемая в апекс-методе, — это операция, которая вычисляет псевдопроекцию, являющуюся обобщением метрической проекции на выпуклое замкнутое множество. Псевдопроекция используется как на стадии Quest, так и на стадии Target. Представлен параллельный алгоритм, использующий фейеровское отображение для вычисления псевдопроекции. Получена аналитическая оценка ресурса параллелизма для этого алгоритма. Также приведен алгоритм, реализующий стадию Target, и доказана его сходимость. Описаны вычислительные эксперименты на кластерной вычислительной системе по применению апекс-метода для решения различных задач линейного программирования.

Ключевые слова: линейное программирование, апекс-метод, итерационный метод, метод проекционного типа, фейеровское отображение, параллельный алгоритм, оценка масштабируемости.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Соколинский Л.Б., Соколинская И.М. О новой версии апекс-метода для решения задач линейного программирования // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2023. Т. 12, № 2. С. 5–46. DOI: 10.14529/cmse230201.

Введение

Данная работа является дальнейшим развитием апекс-метода, предложенного нами в статье [1] для решения задач линейного программирования (ЛП). Актуальность этой темы основывается на следующих факторах. Одним из важных классов приложений ЛП являются нестационарные задачи, связанные с оптимизацией нестационарных процессов [2]. В нестационарных задачах ЛП целевая функция и/или ограничения изменяются в течение вычислительного процесса. В качестве примеров можно привести следующие нестационарные задачи: поддержка принятия решений в высокочастотной торговле [3, 4], задачи гидрогазодинамики [5], оптимальное управление технологическими процессами [6–8], транспортные задачи [9–11], оперативное планирование [12, 13].

Один из стандартных подходов к решению нестационарных задач оптимизации состоит в том, чтобы рассматривать каждое изменение как появление новой задачи оптимизации,

*Работа рекомендована к публикации программным комитетом международной научной конференции «Суперкомпьютерные дни в России 2023».

которую необходимо решать с нуля [2]. Однако такой подход часто непрактичен, поскольку решение проблемы с нуля без повторного использования информации из прошлого может занять слишком много времени. Таким образом, желательно иметь алгоритм оптимизации, способный непрерывно адаптировать решение к изменяющейся среде, повторно используя информацию, полученную в прошлом. Этот подход применим для процессов реального времени, если алгоритм достаточно быстро отслеживает траекторию движения оптимальной точки. В случае больших задач ЛП последнее требует разработки масштабируемых методов и параллельных алгоритмов ЛП.

До сих пор одним из наиболее распространенных способов решения задач ЛП был класс алгоритмов, предложенных и разработанных Данцигом на основе симплекс-метода [14]. Было установлено, что симплексный метод эффективен для решения большого класса задач ЛП. В частности, симплексный метод легко использует преимущества любой гиперразреженности в задачах ЛП [15]. Однако симплекс-метод обладает некоторыми фундаментальными особенностями, которые ограничивают его использование для решения больших задач ЛП. Во-первых, в определенных случаях симплексный метод должен выполнять итерации по всем вершинам симплекса, что соответствует экспоненциальной временной сложности [16–18]. Во-вторых, в большинстве случаев симплекс-метод успешно решает задачи ЛП, содержащие до 50 000 переменных. Однако при решении задач бóльших размерностей часто наблюдается потеря точности ЛП [19], которая не может быть компенсирована даже применением таких мощных вычислительных процедур, как «аффинное масштабирование» или «итеративное уточнение» [20]. В-третьих, в общем случае последовательный характер симплексного метода затрудняет распараллеливание в многопроцессорных системах с распределенной памятью [21]. Были предприняты многочисленные попытки создать масштабируемую параллельную реализацию симплексного метода, но все они оказались безуспешными [22]. Во всех случаях граница масштабируемости составляла от 16 до 32 процессорных узлов (см., например, [23]).

Хачиян доказал [24], используя вариант метода эллипсоидов (предложенный в 1970-х годах Шором [25], Юдиным и Немировским [26]), что задачи ЛП могут быть решены за полиномиальное время. Однако попытки применить этот подход на практике оказались безуспешными, поскольку в подавляющем большинстве случаев метод эллипсоида демонстрировал гораздо худшую эффективность по сравнению с симплекс-методом. Позже Кармаркар [27] показал, что алгоритм внутренних точек, предложенный Дикиным [28], имеет полиномиальную временную сложность и применим на практике. Этот алгоритм породил целую область современных методов внутренних точек [29, 30], которые способны решать большие задачи ЛП с миллионами переменных и миллионами уравнений [31–35]. Более того, эти методы являются самокорректирующимися, а следовательно, обеспечивают высокую точность вычислений. Общим недостатком методов внутренних точек является необходимость найти некоторую допустимую точку, удовлетворяющую всем ограничениям задачи ЛП, перед началом вычислений. Нахождение такой внутренней точки может быть сведено к решению дополнительной задачи ЛП [36]. Еще одним методом нахождения внутренней точки является метод псевдопроекции [37], который использует фейеровские отображения [38]. Другим существенным недостатком метода внутренних точек является его плохая масштабируемость в многопроцессорных системах с распределенной памятью. Существует несколько успешных параллельных реализаций метода внутренних точек для частных случаев (см., например, [39]), но, в общем случае, эффективная параллельная реализация на

многопроцессорных системах для этого метода не может быть построена. В соответствии с этим разработка и исследование новых подходов к решению многомерных нестационарных задач ЛП в режиме реального времени является актуальным направлением.

Одним из наиболее перспективных подходов к решению сложных задач в режиме реального времени является использование нейросетевых моделей [40]. Искусственные нейронные сети — это мощный универсальный инструмент, который применим для решения задач практически во всех областях. Самой популярной моделью нейронной сети является нейронная сеть прямого распространения. Обучение и использование таких сетей могут быть очень эффективно реализованы на графических процессорах [41]. Важным свойством нейронной сети прямого распространения является то, что время решения задачи не зависит от ее параметров. Это свойство необходимо для работы в режиме реального времени. Новаторской работой по использованию нейронных сетей для решения задач ЛП является статья Танка и Хопфилда [42]. В этой статье описывается двухслойная рекуррентная нейронная сеть. Число нейронов в первом слое определяется количеством переменных задачи ЛП. Количество нейронов во втором слое совпадает с количеством ограничений задачи ЛП. Первый и второй слои являются полносвязными. Веса и смещения однозначно определяются коэффициентами и правыми частями линейных неравенств, определяющих ограничения, и коэффициентами линейной целевой функции. Таким образом, эта сеть не требует обучения. Состояние нейронной сети описывается дифференциальным уравнением $\dot{x}(t) = \nabla E(x(t))$, где $E(x(t))$ — энергетическая функция специального типа. Первоначально на вход нейронной сети подается произвольная точка допустимой области. Затем сигнал второго слоя рекурсивно подается на первый слой. В итоге процесс приходит в стабильное состояние, в котором выходной сигнал перестает изменяться. Такое состояние соответствует минимуму энергетической функции, а выходной сигнал является решением задачи ЛП. Подход Танка и Хопфилда был развит и усовершенствован в многочисленных работах (см., например, [43–47]). Основным недостатком этого подхода является непредсказуемое количество рабочих циклов нейронной сети. Следовательно, рекуррентная сеть, основанная на энергетической функции, не может использоваться для решения больших задач ЛП в режиме реального времени.

В недавней статье [48] была предложена n-мерная математическая модель визуализации задач ЛП. Эта модель позволяет использовать нейронные сети прямого распространения, включая сверточные сети [49], для решения многомерных задач ЛП, допустимой областью которых является замкнутое ограниченное непустое множество. Однако в научной литературе практически отсутствуют работы, посвященные использованию сверточных нейронных сетей для решения задач ЛП [50]. Причина в том, что сверточные нейронные сети ориентированы на обработку изображений, но до настоящего времени отсутствовали методы построения обучающих наборов данных, основанные на визуальном представлении многомерных задач ЛП.

В данной статье описывается новый масштабируемый итерационный метод для решения многомерных задач ЛП, получивший название «апекс-метод». Апекс-метод позволяет генерировать обучающие наборы данных для разработки нейронных сетей прямого распространения, способных находить решение многомерной задачи ЛП на основе ее визуального представления. Апекс-метод основан на схеме предиктор/корректор. Предиктор вычисляет точку, принадлежащую допустимой области задачи ЛП. Корректор вычисляет последовательность точек, сходящуюся к точному решению задачи ЛП. Статья организована

следующим образом. В разделе 1 представлен обзор итерационных методов и алгоритмов проекционного типа, ориентированных на решение выпуклых неравенств и задач ЛП. Раздел 2 содержит теоретический базис, используемый в описании апекс-метода. В разделе 3 представлено формализованное описание апекс-метода. Раздел 3.1 посвящен разработке алгоритма построения псевдопроекции и аналитическому исследованию масштабируемости его параллельной версии. В разделе 3.2 описывается стадия Quest. Раздел 3.3 содержит описание стадии Target. В разделе 4 представлены информация о программной реализации апекс-метода и результаты вычислительных экспериментов. В разделе 5 обсуждаются научная и практическая значимость полученных результатов, преимущества и недостатки апекс-метода, и способы его использования. В заключении суммируются представленные в статье результаты и намечаются направления дальнейших исследований. В конце статьи приведены основные Обозначения, используемые при описании апекс-метода.

1. Обзор работ по итерационным методам проекционного типа

В этом разделе представлен обзор работ, посвященных итерационным методам проекционного типа, используемым для решения задач совместности выпуклых неравенств и задач ЛП. Задача совместности (допустимости) выпуклых неравенств заключается в нахождении некоторого решения системы выпуклых неравенств. Эта задача возникает в многочисленных приложениях, таких как статистика, параметрическое оценивание, распознавание образов, восстановление изображений, томография и других [51]. В случае линейных неравенств задача совместности может быть сформулирована следующим образом. Имеется система линейных неравенств в матричном виде:

$$Ax \leqslant b, \tag{1}$$

где $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^n$. Во избежание вырожденности будем предполагать, что $m > 1$. Задача линейной совместности, заключается в нахождении точки $\tilde{x} \in \mathbb{R}^n$, удовлетворяющей матричному неравенству (1). Везде далее мы будем предполагать, что такая точка существует, то есть система (1) является совместной.

Методы проекционного типа основаны на следующей геометрической интерпретации задачи линейной совместности. Обозначим через $a_i \in \mathbb{R}^n$ вектор, состоящий из элементов i-той строки матрицы A. Тогда матричное неравенство $Ax \leqslant b$ может быть представлено в виде системы неравенств

$$\langle a_i, x \rangle \leqslant b_i, i = 1, \ldots, m. \tag{2}$$

Здесь $\langle \cdot, \cdot \rangle$ обозначает скалярное произведение двух векторов. Везде далее мы предполагаем, что

$$a_i \neq \mathbf{0} \tag{3}$$

для всех $i = 1, \ldots, m$. Каждое неравенство $\langle a_i, x \rangle \leqslant b_i$ определяет замкнутое полупространство

$$\hat{H}_i = \{x \in \mathbb{R}^n | \langle a_i, x \rangle \leqslant b_i\} \tag{4}$$

и ограничивающую его гиперплоскость

$$H_i = \{x \in \mathbb{R}^n | \langle a_i, x \rangle = b_i\}. \tag{5}$$

Для любой точки $x \in \mathbb{R}^n$ ортогональная проекция $\pi(x)$ на гиперплоскость H_i может быть вычислена по формуле

$$\pi_i(x) = x - \frac{\langle a_i, x\rangle - b_i}{\|a_i\|^2} a_i. \tag{6}$$

Здесь и далее $\|\cdot\|$ обозначает евклидову норму. Определим допустимый многогранник

$$M = \bigcap_{i=1}^{m} \hat{H}_i, \tag{7}$$

представляющий множество допустимых точек системы (1). Заметим, что M в этом случае будет замкнутым выпуклым множеством. Мы будем предполагать, что $M \neq \emptyset$, то есть система (1) имеет решение. С геометрической точки зрения задача линейной совместности состоит в нахождении точки $\tilde{x} \in M$.

Первыми работами, посвященными задаче линейной совместности, были работы Качмарца (Kaczmarz) и Чиммино (Cimmino). Качмарц в работе [52] (английский перевод [53]) предложил следующий метод последовательных проекций для решения совместной системы линейных неравенств

$$\langle a_i, x\rangle = b_i, i = 1, \ldots, m. \tag{8}$$

Начиная с произвольной точки $x^{(0,m)} \in \mathbb{R}$, этот метод строит последовательность групп точек

$$x^{(k,1)} = \pi_1\left(x^{(k-1,m)}\right), x^{(k,2)} = \pi_2\left(x^{(k,1)}\right), \ldots, x^{(k,m)} = \pi_m\left(x^{(k,m-1)}\right) \tag{9}$$

для $k = 1, 2, 3, \ldots$ Здесь π_i $(i = 1, \ldots, m)$ обозначает ортогональную проекцию на гиперплоскость H_i, вычисляемую по формуле (6). Качмарц показал, что последовательность (9) сходится к решению системы (8). С геометрической точки зрения метод Качмарца может быть описан следующим образом. На первом шаге строится ортогональная проекция начальной точки $x^{(0,m)}$ на гиперплоскость H_1. Полученная точка $x^{(1,1)}$, в свою очередь, проецируется на H_2, что дает нам точку $x^{(1,2)}$. Точка $x^{(1,2)}$ проецируется на H_3, что дает нам точку $x^{(1,3)}$, и так далее. Последней точкой в первой группе будет точка $x^{(1,m)}$, получающаяся в результате ортогональной проекция точки $x^{(1,m-1)}$ на гиперплоскость H_m. Вторая группа точек строится аналогичным образом, используя в качестве начальной точку $x^{(1,m)}$. Процесс продолжается для $k = 3, 4, 5 \ldots$

Чиммино в [54] (английское описание [55]) предложил метод одновременных проекций для решения задачи линейной совместности. В своем методе вместо ортогональных проекций Чиммино использует ортогональные отражения, вычисляемые по формуле

$$\rho_i(x) = x - 2\frac{\langle a_i, x\rangle - b_i}{\|a_i\|^2} a_i. \tag{10}$$

Ортогональное отражение строит точку $\rho_i(x)$, симметричную точке x относительно гиперплоскости H_i. Для текущего приближения $x^{(k)}$ метод Чиммино вычисляет ортогональные отражения сразу относительно всех гиперплоскостей H_i $(i = 1, \ldots, m)$ и затем использует выпуклую комбинацию полученных точек для формирования следующего приближения:

$$x^{(k+1)} = \sum_{i=1}^{m} w_i \rho_i\left(x^{(k)}\right), \tag{11}$$

где $w_i > 0$ $(i = 1, \ldots, m)$, $\sum\limits_{i=1}^{m} w_i = 1$. При $w_i = \frac{1}{m}$ $(i = 1, \ldots, m)$ формула (11) принимает вид

$$x^{(k+1)} = \frac{1}{m} \sum_{i=1}^{m} \rho_i \left(x^{(k)} \right). \tag{12}$$

Агмон (Agmon) [56], Моцкин (Motzkin), Шенберг (Schoenberg) [57] предложили релаксационный метод, являющийся обобщением проекционного метода Качмарца на случай линейных неравенств. Для решения системы (1) они используют следующее релаксационное отображение:

$$\pi_i^{\lambda}(x) = (1 - \lambda)x + \lambda \pi_i(x), \tag{13}$$

где $0 < \lambda < 2$. Очевидно, что $\pi_i^1(x) = \pi_i(x)$, то есть при $\lambda = 1$ релаксационное отображение превращается в ортогональную проекцию. Для вычисления следующего приближения релаксационный метод использует формулу

$$x^{(k+1)} = \pi_l^{\lambda} \left(x^{(k)} \right), \tag{14}$$

где

$$l = \arg\max_i \left\{ \left\| x^{(k)} - \pi_i \left(x^{(k)} \right) \right\| \middle| \, x^{(k)} \notin \hat{H}_i \right\}. \tag{15}$$

С неформальной точки зрения, следующее приближение $x^{(k+1)}$ получается в результате релаксационного отображения предыдущего приближения $x^{(k)}$ относительно самой дальней гиперплоскости H_l, ограничивающей полупространство $\hat{H}_l$, не содержащее точку $x^{(k)}$. Агмон в [56] показал, что последовательность $x^{(k)}$ сходится к граничной точке допустимого многогранника M.

Цензор (Censor) и Эльфвинг (Elfving) в [58] обобщили метод Чиммино на случай линейных неравенств. Они рассматривают ослабленную (relaxed) проекцию на полупространство, определяемую формулой

$$\hat{\pi}_i^{\lambda}(x) = (1 - \lambda)x - \lambda \frac{\max\{0, \langle a_i, x \rangle - b_i\}}{\|a_i\|^2} a_i, \tag{16}$$

и получают следующее итерационное уравнение

$$x^{(k+1)} = \sum_{i=1}^{m} w_i \hat{\pi}_i^{\lambda} \left(x^{(k)} \right). \tag{17}$$

Здесь $0 < \lambda < 2$, $w_i > 0$ $(i = 1, \ldots, m)$, $\sum\limits_{i=1}^{m} w_i = 1$. Де Пьеро (De Pierro) в [59] предложил схему доказательства сходимости этого метода, отличающуюся от схемы цензора и Эльфвинга. Подход де Пьеро также применим для случая, когда исходная система линейных неравенств несовместна. В этом случае при $\lambda = 1$ последовательность (17) сходится к точке минимума функции $f(x) = \sum_{i=1}^{m} w_i \|\hat{\pi}_i(x) - x\|^2$, являющейся взвешенным (с весами w_i) решением системы (1) методом наименьших квадратов.

Проекционные методы, основанные на подходе Чиммино, допускают эффективное распараллеливание, поскольку ортогональные проекции/отражения могут вычисляться одновременно и независимо. Масштабируемость метода Чиммино на многопроцессорных системах с распределенной памятью была исследована в работе [60]. Применимость проекционных методов по схеме Чиммино для решения нестационарных систем линейных неравенств рассматривалась в работе [61].

Решение систем линейных неравенств тесно связано с задачами ЛП, поэтому методы проекционного типа могут быть эффективно использованы для решения этого класса задач. Эквивалентность задачи линейной совместности и задачи ЛП основана на прямо-двойственном методе решения задачи ЛП. Рассмотрим прямую задачу ЛП в матричной форме:

$$\bar{x} = \arg\max_{x} \left\{ \langle c, x \rangle \,|\, Ax \leqslant b, x \geqslant \mathbf{0} \right\}, \tag{18}$$

где $c, x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$, $c \neq \mathbf{0}$. Сформулируем двойственную задачу по отношению к задаче (18):

$$\bar{u} = \arg\min_{u} \left\{ \langle b, u \rangle \,|\, A^T u \geqslant c, u \geqslant \mathbf{0} \right\}, \tag{19}$$

где $u \in \mathbb{R}^m$. Для прямой и двойственной задач ЛП справедливо следующее равенство:

$$\langle c, \bar{x} \rangle = \max_{Ax \leqslant b, x \geqslant \mathbf{0}} \langle c, x \rangle = \min_{A^T u \geqslant c, u \geqslant \mathbf{0}} \langle b, u \rangle = \langle b, \bar{u} \rangle . \tag{20}$$

Ерёмин в [38, 62] предложил следующий метод решения задачи ЛП, основанный на прямо-двойственном подходе. Пусть система линейных неравенств

$$A'x \leqslant b' \tag{21}$$

задает допустимую область прямой задачи (18). Указанная система получается из системы $Ax \leqslant b$ путем добавления векторного неравенства $-x \leqslant \mathbf{0}$. В данном случае $A' \in \mathbb{R}^{(m+n) \times n}$ и $b' \in \mathbb{R}^{m+n}$. Пусть a'_i обозначает i-тую строку матрицы A'. Сопоставим каждому неравенству $\langle a'_i, x \rangle \leqslant b'_i$ закрытое полупространство

$$\hat{H}'_i = \left\{ x \in \mathbb{R}^n \middle| \left\langle a'_i, x \right\rangle \leqslant b'_i \right\}, \tag{22}$$

и ограничивающую его гиперплоскость

$$H'_i = \left\{ x \in \mathbb{R}^n \middle| \left\langle a'_i, x \right\rangle = b'_i \right\}. \tag{23}$$

Обозначим через $\pi'_i(x)$ ортогональную проекцию точки x на гиперплоскость H'_i:

$$\pi'_i(x) = x - \frac{\langle a'_i, x \rangle - b'_i}{\|a'_i\|^2} a'_i. \tag{24}$$

Определим проекцию на полупространство $\hat{H}'_i$:

$$\hat{\pi}'_i(x) = x - \frac{\max\{0, \langle a'_i, x \rangle - b'_i\}}{\|a'_i\|^2} a'_i. \tag{25}$$

Указанная проекция обладает следующими двумя свойствами:

$$x \notin \hat{H}'_i \Rightarrow \hat{\pi}'_i(x) = \pi'_i(x); \tag{26}$$

$$x \in \hat{H}'_i \Rightarrow \hat{\pi}'_i(x) = x. \tag{27}$$

Определим отображение $\varphi_1 : \mathbb{R}^n \to \mathbb{R}^n$ следующим образом:

$$\varphi_1(x) = \frac{1}{m+n} \sum_{i=1}^{m+n} \hat{\pi}'_i(x). \tag{28}$$

Аналогичным образом определим допустимую область двойственной задачи (19):

$$D'x \geqslant c', \tag{29}$$

где $D = A^T \in \mathbb{R}^{n\times m}$, $D' \in \mathbb{R}^{(m+n)\times m}$, $c' \in \mathbb{R}^{n+m}$. Обозначим

$$\hat{\eta}'_j(u) = u - \frac{\max\left\{0, \left\langle d'_j, u\right\rangle - c'_j\right\}}{\left\|d'_j\right\|^2} d'_j, \tag{30}$$

и определим отображение $\varphi_2 : \mathbb{R}^m \to \mathbb{R}^m$ следующим образом:

$$\varphi_2(u) = \frac{1}{n+m}\sum_{j=1}^{n+m} \hat{\eta}'_j(x). \tag{31}$$

Далее, определим отображение $\varphi_3 : \mathbb{R}^{n+m} \to \mathbb{R}^{n+m}$, соответствующее равенству (20):

$$\varphi_3\left([x,u]\right) = [x,u] - \frac{\langle c,x\rangle - \langle b,u\rangle}{\|c\|^2 + \|b\|^2}[c,-b]. \tag{32}$$

Здесь $[\cdot\,,\cdot]$ обозначает конкатенацию двух векторов.

Наконец, определим $\varphi : \mathbb{R}^{n+m} \to \mathbb{R}^{n+m}$ следующим образом:

$$\varphi\left([x,u]\right) = \varphi_3\left([\varphi_1(x), \varphi_2(u)]\right). \tag{33}$$

Если допустимая область прямой задачи является ограниченным непустым множеством, то последовательность, задаваемая формулой

$$\left[x^{(k+1)}, u^{(k+1)}\right] = \varphi\left(\left[x^{(k)}, u^{(k)}\right]\right), \tag{34}$$

будет сходиться к точке $[\bar{x}, \bar{u}]$, где $\bar{x}$ является решением прямой задачи (18), а $\bar{u}$ является решением двойственной задачи (19).

В статье [63] предлагается метод решения невырожденных задач ЛП, основанный на вычислении ортогональной проекции некоторой специальной точки, не зависящей от основной части исходных данных, описывающих задачу ЛП, на проблемно-зависимый конус, порождаемый ограничивающими неравенствами. Фактически, этот метод решает симметричную положительно определенную систему линейных уравнений специального вида. Автор описывает конечный алгоритм на основе метода активных множеств, способный вычислять ортогональные проекции для задач с тысячами строк и столбцов. Основным недостатком этого метода является существенное увеличение размерности исходной задачи.

Цензор в работе [64] описывает применение метода линейной супериоризации (LinSup) для решения задач ЛП. Метод LinSup направляет используемый итерационный алгоритм проекционного типа в сторону точек с увеличивающимся значением целевой функции. При этом LinSup не гарантирует нахождение точного оптимума задачи ЛП. Этот процесс не идентичен тому, который используется ЛП-решателями, но это возможная альтернатива симплекс-методу для решения задач очень большого размера. Основная идея LinSup состоит в том, чтобы добавить дополнительный терм, называемый возмущающим термом, в итерационное уравнение проекционного метода. Возмущающий терм направляет алгоритм

поиска допустимой точки в сторону увеличения значения целевой функции. В контексте задачи ЛП (18) целевая функция имеет вид $f(x) = \langle c, x\rangle$, и LinSup добавляет в итерационное уравнение (17) возмущающий терм вида $\left(-\eta\frac{c}{\|c\|}\right)$:

$$x^{(k+1)} = \left(-\eta\frac{c}{\|c\|}\right) + \sum_{i=1}^{m} w_i\hat{\pi}_i^{\lambda}\left(x^{(k)}\right). \tag{35}$$

Здесь $0 < \eta < 1$ — величина возмущения, являющаяся настраиваемым параметром алгоритма.

В статье [48] предлагается математическая модель для визуального представления многомерных задач ЛП. Для визуализации задачи ЛП вводится целевая гиперплоскость H_c, нормаль к которой совпадает с градиентом целевой функции $f(x) = \langle c, x\rangle$. В случае поиска максимума целевая гиперплоскость располагается так, чтобы значение целевой функции во всех ее точках было больше значения целевой функции в любой точке допустимого многогранника M. Для любой точки $g \in H_c$ определяется целевая проекция этой точки на многогранник M в соответствии со следующей формулой:

$$\gamma_M(g) = \begin{cases} \arg\min\limits_{x} \{\, \|x - g\| \,|\, x \in M, \pi_{H_c}(x) = g\}\,, & \text{если } \exists x \in M : \pi_{H_c}(x) = g; \\ +\infty, & \text{если } \neg\exists x \in M : \pi_{H_c}(x) = g. \end{cases} \tag{36}$$

Здесь, $\pi_{H_c}(x)$ обозначает ортогональную проекцию точки x на гиперплоскость H_c. На целевой гиперплоскости H_c строится прямоугольная решетка точек $\mathfrak{G} \in \mathbb{R}^n \times \mathbb{R}^{K^{(n-1)}}$, где K — число точек по одному измерению. Каждой точке $g \in \mathfrak{G}$ сопоставляется вещественное число $\|\gamma_M(g) - g\|$. В результате получается матрица размерности $(n-1)$, представляющая собой образ задачи ЛП. Этот подход открывает возможность использования нейронных сетей прямого распространения, включая сверточные, для решения многомерных задач ЛП. Основной проблемой для реализации такого подхода на практике является проблема построения обучающего набора данных. Обзор литературы показывает, что в настоящее время не существует методов, позволяющих построить обучающий набор данных, совместимый с представленным подходом. В следующих разделах мы опишем такой метод.

2. Теоретический базис

Данный раздел содержит необходимый теоретических базис, используемый для описания апекс-метода. Рассмотрим задачу ЛП в следующем виде:

$$\bar{x} = \arg\max_{x\in\mathbb{R}^n} \{\langle c, x\rangle \,|\, Ax \leqslant b\}\,, \tag{37}$$

где $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m\times n}$, $m > 1$, $c \neq \mathbf{0}$. Мы предполагаем, что ограничение

$$-x \leqslant \mathbf{0} \tag{38}$$

также включено в матричное неравенство $Ax \leqslant b$. Обозначим через $\mathcal{P}$ множество индексов, нумерующих строки матрицы A:

$$\mathcal{P} = \{1, \cdots, m\}\,. \tag{39}$$

Пусть $a_i \in \mathbb{R}^n$ обозначает вектор, представляющий i-тую строку матрицы A. Мы предполагаем, что $a_i \neq \mathbf{0}$ для всех $i \in \mathcal{P}$. Обозначим через $\hat{H}_i$ замкнутое полупространство,

определяемое неравенством $\langle a_i, x\rangle \leqslant b_i$, а через H_i — его ограничивающую гиперплоскость:

$$\hat{H}_i = \{x \in \mathbb{R}^n | \langle a_i, x\rangle \leqslant b_i\}; \tag{40}$$

$$H_i = \{x \in \mathbb{R}^n | \langle a_i, x\rangle = b_i\}. \tag{41}$$

Определение 1. Полупространство $\hat{H}_i$ называется доминантным, если

$$\forall x \in \hat{H}_i, \forall \lambda \in \mathbb{R}_{>0} : x + \lambda c \in \hat{H}_i. \tag{42}$$

Геометрический смысл данного определения состоит в том, что луч, исходящий из любой точки доминантного полупространства в направлении вектора c, принадлежит этому полупространству.

Определение 2. Полупространство $\hat{H}_i$ называется рецессивным, если оно не является доминантным, то есть

$$\forall x \in \hat{H}_i, \exists \lambda \in \mathbb{R}_{>0} : x + \lambda c \notin \hat{H}_i. \tag{43}$$

Геометрический смысл этого определения состоит в том, что луч, исходящий из любой точки рецессивного полупространства в направлении вектора c, выходит за пределы этого полупространства.

Утверждение 1. Следующее условие является необходимым и достаточным для того, чтобы полупространство $\hat{H}_i$ было рецессивным:

$$\langle a_i, c\rangle > 0. \tag{44}$$

Доказательство. Сначала докажем необходимость. Пусть условие (43) имеет место. Обозначим

$$x' = \frac{\beta a_i}{\|a_i\|^2}. \tag{45}$$

Имеем

$$\langle a_i, x'\rangle = \left\langle a_i, \frac{\beta a_i}{\|a_i\|^2}\right\rangle = \beta\frac{\langle a_i, a_i\rangle}{\|a_i\|^2} = \beta, \tag{46}$$

то есть $x' \in \hat{H}_i$ в силу (40). Согласно условию (43) существует $\lambda' \in \mathbb{R}_{>0}$ такое, что

$$x' + \lambda' c \notin \hat{H}_i, \tag{47}$$

то есть

$$\langle a_i, x' + \lambda' c\rangle > \beta. \tag{48}$$

Подставляя сюда правую часть равенства (45) вместо x', получаем

$$\left\langle a_i, \frac{\beta a_i}{\|a_i\|^2} + \lambda' c\right\rangle > \beta. \tag{49}$$

Поскольку $\lambda' > 0$, отсюда следует, что

$$\langle a_i, c\rangle > 0. \tag{50}$$

Это доказывает необходимость.

Теперь докажем достаточность. Пусть условие (44) имеет место, но полупространство $\hat{H}_i$ при этом не является рецессивным, то есть

$$\forall x \in \hat{H}_i, \forall \lambda \in \mathbb{R}_{>0} : x + \lambda c \in \hat{H}_i. \tag{51}$$

Так как x', вычисляемый по формуле (45), принадлежит $\hat{H}_i$, отсюда следует

$$x' + \lambda c \in \hat{H}_i \tag{52}$$

для всех $\lambda \in \mathbb{R}_{>0}$, что равносильно

$$\langle a_i, x' + \lambda c \rangle \leqslant \beta. \tag{53}$$

Поставляя сюда правую часть равенства (45) вместо x', получаем

$$\left\langle a_i, \frac{\beta a_i}{\|a_i\|^2} + \lambda c \right\rangle \leqslant \beta. \tag{54}$$

Поскольку $\lambda > 0$, отсюда следует

$$\langle a_i, c \rangle \leqslant 0. \tag{55}$$

Получили противоречие с (44). Таким образом, достаточность также доказана. □

Обозначим

$$e_c = \frac{c}{\|c\|}. \tag{56}$$

Другими словами, e_c обозначает единичный вектор, сонаправленный с вектором c.

Утверждение 2. Пусть полупространство $\hat{H}_i$ является рецессивным. Тогда для любой точки $x' \in \mathbb{R}^n$ и любого положительного числа $\eta > 0$ точка

$$z = x' + \left(\eta + \frac{b_i - \langle a_i, x' \rangle}{\langle a_i, e_c \rangle} \right) e_c \tag{57}$$

не принадлежит полупространству $\hat{H}_i$, то есть

$$\langle a_i, z \rangle > b_i. \tag{58}$$

Доказательство. Так как полупространство $\hat{H}_i$ является рецессивным, то в соответствии с утверждением 1 справедливо следующее неравенство:

$$\langle a_i, c \rangle > 0. \tag{59}$$

В силу (57) мы имеем

$$\langle a_i, z \rangle = \left\langle a_i, x' + \left(\eta + \frac{b_i - \langle a_i, x' \rangle}{\langle a_i, e_c \rangle} \right) e_c \right\rangle = \eta \langle a_i, e_c \rangle + b_i. \tag{60}$$

Подставляя в (60) правую часть равенства (56) вместо e_c, получаем

$$\langle a_i, z \rangle = \frac{\eta}{\|c\|} \langle a_i, c \rangle + b_i. \tag{61}$$

Поскольку $\eta > 1$, из (59) следует $\frac{\eta}{\|c\|} \langle a_i, c \rangle > 0$. Это означает, что из (61) следует $\langle a_i, z \rangle > b_i$, то есть $z \notin \hat{H}_i$. *Утверждение доказано.* □

Определим

$$\mathcal{I} = \{ i \in \mathcal{P} \,|\langle a_i, c\rangle > 0 \}, \tag{62}$$

то есть $\mathcal{I}$ представляет множество индексов, для которых полупространство $\hat{H}_i$ является рецессивным. Поскольку допустимый многогранник M представляет собой ограниченное множество, имеем

$$\mathcal{I} \neq \emptyset. \tag{63}$$

Следствие 1. Пусть имеется произвольная допустимая точка x' задачи ЛП (37):

$$\forall i \in \mathcal{P} : \left\langle a_i, x' \right\rangle \leqslant b_i. \tag{64}$$

Тогда для любого положительного числа $\eta \in \mathbb{R}_{>0}$ точка

$$z = x' + \left(\eta + \max \left\{ \left. \frac{b_i - \langle a_i, x'\rangle}{\langle a_i, e_c\rangle} \right| i \in \mathcal{I} \right\} \right) e_c \tag{65}$$

не принадлежит ни одному рецессивному пространству $\hat{H}_i$, то есть

$$\forall i \in \mathcal{I} : \langle a_i, z\rangle > b_i. \tag{66}$$

Доказательство. Условие (64) равносильно условию

$$\forall i \in \mathcal{I} : b_i - \left\langle a_i, x' \right\rangle \geqslant 0. \tag{67}$$

Из (62) и (56) получаем

$$\forall i \in \mathcal{I} : \langle a_i, e_c\rangle > 0. \tag{68}$$

Отсюда с учетом (67) следует, что

$$\max \left\{ \left. \frac{b_i - \langle a_i, x'\rangle}{\langle a_i, e_c\rangle} \right| i \in \mathcal{I} \right\} \geqslant 0 \tag{69}$$

для всех $i \in \mathcal{I}$. Зафиксируем произвольный $j \in \mathcal{I}$ и определим

$$\eta' = \eta + \max \left\{ \left. \frac{b_i - \langle a_i, x'\rangle}{\langle a_i, e_c\rangle} \right| i \in \mathcal{I} \right\} - \frac{b_j - \langle a_j, x'\rangle}{\langle a_j, e_c\rangle}, \tag{70}$$

где $\eta > 0$. Принимая во внимание (69), отсюда следует $\eta' > 0$. Из (65) и (70) получаем

$$z = x' + \left(\eta + \max \left\{ \left. \frac{b_i - \langle a_i, x'\rangle}{\langle a_i, e_c\rangle} \right| i \in \mathcal{I} \right\} \right) e_c = x' + \left(\eta' + \frac{b_j - \langle a_j, x'\rangle}{\langle a_j, e_c\rangle} \right) e_c. \tag{71}$$

В силу утверждения 2 это означает, что $\langle a_j, z\rangle > b_j$, то есть точка z, вычисляемая по формуле (65), не принадлежит полупространству $\hat{H}_j$ для всех $j \in \mathcal{I}$. Следствие доказано. □

Следующее утверждение определяет область, где может находиться решение задачи ЛП (37).

Утверждение 3. Пусть $\bar{x}$ является решением задачи ЛП (37). Тогда найдется индекс $i' \in \mathcal{I}$ такой, что

$$\bar{x} \in H_{i'}, \tag{72}$$

то есть существует рецессивное полупространство $\hat{H}_{i'}$ такое, что ограничивающая его гиперплоскость $H_{i'}$ содержит $\bar{x}$.

Доказательство. Обозначим через $\mathcal{J}$ множество индексов, для которых полупространство $\hat{H}_j$ является доминантным:

$$\mathcal{J} = \mathcal{P} \backslash \mathcal{I}. \tag{73}$$

Так как $\bar{x}$ принадлежит допустимой области задачи ЛП (37), справедливы следующие включения:

$$\bar{x} \in \bigcap_{j \in \mathcal{J}} \hat{H}_j, \tag{74}$$

$$\bar{x} \in \bigcap_{i \in \mathcal{I}} \hat{H}_i. \tag{75}$$

Определим луч Y следующим образом:

$$Y = \{\bar{x} + \lambda c \,|\lambda \in \mathbb{R}_{\geqslant 0}\}. \tag{76}$$

В соответствии с определением 1 имеем

$$Y \subset \bigcap_{j \in \mathcal{J}} \hat{H}_j, \tag{77}$$

то есть луч Y принадлежит всем доминантным полупространствам. В силу определения 2

$$\forall i \in \mathcal{I}, \exists \lambda \in \mathbb{R}_{>0} \, : \bar{x} + \lambda c \notin \hat{H}_i. \tag{78}$$

Принимая во внимание (75), это означает, что

$$\forall i \in \mathcal{I} : Y \cap H_i = y_i \in \mathbb{R}^n, \tag{79}$$

то есть луч Y пересекает любую гиперплоскость H_i, ограничивающую рецессивное полупространство $\hat{H}_i$, в единственной точке $y_i \in \mathbb{R}^n$. Положим

$$i' = \arg\min_{i \in \mathcal{I}} \{\|\bar{x} - y_i\| \,|y_i = Y \cap H_i\}, \tag{80}$$

то есть гиперплоскость $H_{i'}$ является ближайшей к точке $\bar{x}$ для всех $i \in \mathcal{I}$. Обозначим через $\bar{y}$ пересечение луча Y и гиперплоскости $H_{i'}$:

$$\bar{y} = Y \cap H_{i'}. \tag{81}$$

В соответствии с (75), (76) и (80)

$$\bar{y} \in \bigcap_{i \in \mathcal{I}} \hat{H}_i, \tag{82}$$

то есть точка $\bar{y}$ принадлежит всем рецессивным полупространствам. В силу (77) отсюда следует, что

$$\bar{y} \in \bigcap_{i \in \mathcal{P}} \hat{H}_i. \tag{83}$$

Это означает, что $\bar{y}$ принадлежит допустимой области задачи ЛП (37).

Положим

$$\lambda' = \|\bar{x} - \bar{y}\|. \tag{84}$$

Тогда в силу (76) имеем

$$\langle c, \bar{y} \rangle = \langle c, \bar{x} + \lambda' e_c \rangle = \langle c, \bar{x} \rangle + \lambda' \frac{\langle c, c \rangle}{\|c\|} = \langle c, \bar{x} \rangle + \lambda' \|c\| . \tag{85}$$

Поскольку $\bar{x}$ является решением задачи ЛП (37), следующее условие имеет место:

$$\forall y \in \bigcap_{i \in \mathcal{P}} \hat{H}_i : \langle c, y \rangle \leqslant \langle c, \bar{x} \rangle . \tag{86}$$

Сопоставляя это с (83), получаем

$$\langle c, \bar{y} \rangle \leqslant \langle c, \bar{x} \rangle . \tag{87}$$

Принимая во внимание, что $\lambda' \geqslant 0$ и $c \neq \mathbf{0}$, в силу (85) и (87) имеем $\lambda' = 0$. В соответствии с (84) отсюда следует, что $\bar{x} = \bar{y}$. В силу (81) это означает, что $\bar{x} \in H_{i'}$, где $\hat{H}_{i'}$ является рецессивным полупространством. *Утверждение доказано.* □

Определение 3. Пусть $M \neq \emptyset$ — выпуклое замкнутое множество. Однозначное отображение $\varphi : \mathbb{R}^n \to \mathbb{R}^n$ называется M-фейеровским отображением [38], если

$$\forall x \in M : \varphi(x) = x, \tag{88}$$

и

$$\forall x \notin M, \forall y \in \mathbb{R}^n : \|\varphi(x) - y\| < \|x - y\|. \tag{89}$$

Утверждение 4. Пусть $M \neq \emptyset$ — выпуклое замкнутое множество, $x^{(0)}$ — произвольная точка в $\mathbb{R}^n$. Если $\varphi(\cdot)$ является непрерывным M-фейеровским отображением, то последовательность

$$\left\{ x^{(k)} = \varphi^k \left(x^{(0)} \right) \right\}_{k=1}^{\infty},$$

порождаемая этим отображением, сходится к точке, принадлежащей M:

$$x^{(k)} \to \tilde{x} \in M. \tag{90}$$

Доказательство. Сходимость непосредственно следует из теоремы 6.1 и следствия 6.1 в [38]. *Утверждение доказано.* □

Обозначим через $\pi_i(x)$ ортогональную проекцию точки x на гиперплоскость H_i:

$$\pi_i(x) = x - \frac{\langle a_i, x \rangle - b_i}{\|a_i\|^2} a_i. \tag{91}$$

Следующее утверждение дает нам непрерывное M-фейеровское отображение, которое будет использоваться в апекс-методе.

Утверждение 5. Пусть $M \neq \emptyset$ — допустимый многогранник задачи ЛП (37):

$$M = \bigcap_{i=1}^{m} \hat{H}_i. \tag{92}$$

Известно, что в этом случае M является выпуклым замкнутым множеством. Для произвольной точки $x \in \mathbb{R}^n$ определим множество индексов

$$\mathcal{J}_x = \{ i \,|\langle a_i, x \rangle > b_i; i \in \mathcal{P} \} . \tag{93}$$

Другими словами, $\mathcal{J}_x$ — множество индексов полупространств $\hat{H}_i$, которые не содержат точку x. Однозначное отображение $\psi : \mathbb{R}^n \to \mathbb{R}^n$, задаваемое формулой

$$\psi(x) = \begin{cases} x, & \text{если } x \in M; \\ \frac{1}{|\mathcal{J}_x|} \sum\limits_{i \in \mathcal{J}_x} \pi_i(x), & \text{если } x \notin M, \end{cases} \tag{94}$$

является непрерывным M-фейеровским отображением.

Доказательство. Очевидно, что отображение $\psi(\cdot)$ является непрерывным. Покажем, что выполняется условие (89). Доказательство проведем по общей схеме, представленной в [38]. Пусть $y \in M$ и $x \notin M$. Это означает, что

$$\mathcal{J}_x \neq \emptyset. \tag{95}$$

В силу (93) для всех $i \in \mathcal{J}_x$ справедливо неравенство

$$\|\pi_i(x) - x\| > 0. \tag{96}$$

Согласно лемме 3.2 в [38] для всех $i \in \mathcal{J}_x$ также выполняется следующее неравенство:

$$\|\pi_i(x) - y\|^2 \leqslant \|x - y\|^2 - \|\pi_i(x) - x\|^2. \tag{97}$$

Отсюда следует

$$\begin{aligned} &\|y - \psi(x)\|^2 = \left\| y - \tfrac{1}{|\mathcal{J}_x|} \sum_{i \in \mathcal{J}_x} \pi_i(x) \right\|^2 = \left\| \tfrac{1}{|\mathcal{J}_x|} \sum_{i \in \mathcal{J}_x} (y - \pi_i(x)) \right\|^2 \leqslant \tfrac{1}{|\mathcal{J}_x|^2} \sum_{i \in \mathcal{J}_x} \|y - \pi_i(x)\|^2 \leqslant \\ &\leqslant \tfrac{1}{|\mathcal{J}_x|} \sum_{i \in \mathcal{J}_x} \|y - \pi_i(x)\|^2 \leqslant \tfrac{1}{|\mathcal{J}_x|} \sum_{i \in \mathcal{J}_x} \left(\|x - y\|^2 - \|\pi_i(x) - x\|^2 \right) \leqslant \\ &\leqslant \|x - y\|^2 - \tfrac{1}{|\mathcal{J}_x|} \sum_{i \in \mathcal{J}_x} \|\pi_i(x) - x\|^2. \end{aligned}$$

В соответствии с (95) и (96) следующее неравенство имеет место:

$$\tfrac{1}{|\mathcal{J}_x|} \sum_{i \in \mathcal{J}_x} \|\pi_i(x) - x\|^2 > 0. \tag{98}$$

Отсюда

$$\forall x \notin M, \forall y \in \mathbb{R}^n : \|\psi(x) - y\| < \|x - y\|.$$

Утверждение доказано. □

Определение 4. Пусть $M \neq \emptyset$ — допустимый многогранник задачи ЛП (37), $\psi(\cdot)$ — отображение, определяемое формулой (94). *Псевдопроекцией* $\rho_M(x)$ точки x на допустимый многогранник M называется предельная точка последовательности $[x, \psi(x), \psi^2(x), \ldots, \psi^k(x), \ldots]$:

$$\lim_{k \to \infty} \left\| \rho_M(x) - \psi^k(x) \right\| = 0. \tag{99}$$

Корректность этого определения вытекает из утверждений 4 и 5.

3. Описание апекс-метода

В этом разделе мы опишем новый масштабируемый итерационный метод решения задачи ЛП (37), получивший название «апекс-метод». Апекс-метод построен по схеме предиктор/корректор и включает в себя две последовательные стадии: Quest (предиктор) и Target (корректор). Стадия Quest находит грубое начальное приближение для задачи ЛП (37). Стадия Target уточняет это начальное приближение с определенной точностью. Основной операцией, используемой как на стадии Quest, так и на стадии Target, является операция вычисления псевдопроекции (см. определение 4). Следующий раздел посвящен описанию и исследованию алгоритма вычисления псевдопроекции.

3.1. Алгоритм вычисления псевдопроекции

Базовой операцией, используемой в апекс-методе, является операция псевдопроектирования, заключающаяся в последовательном применении отображения $\psi(\cdot)$, задаваемого формулой (94), к исходной точке. В данном разделе мы рассмотрим реализацию операции псевдопроектирования в виде последовательного и параллельного алгоритмов. Согласно определению 4 операция псевдопроектирования $\rho_M(\cdot)$ отображает произвольную точку $x \in \mathbb{R}^n$ в точку $\rho_M(x)$, принадлежащую допустимому многограннику M, представляющему допустимую область задачи ЛП (37). Вычисление $\rho_M(x)$ организуется в виде итерационного процесса с использованием формулы (94). Последовательная реализация этого процесса представлена в виде алгоритма 1. Кратко прокомментируем шаги этого алгоритма. Основной итерационный процесс, вычисляющий последовательность фейеровских приближений, представлен в виде цикла **repeat–until** (шаги 4–20). На шагах 5–10 строится множество $\mathcal{J}$, содержащее индексы полупространств $\hat{H}_i$, которым не принадлежит текущее приближение $x^{(k)}$. На шагах 14–18 вычисляется следующее приближение $x^{(k+1)}$ по формуле (94). Алгоритм завершает свою работу, когда расстояние между соседними приближения станет меньше малой положительной константы ϵ.

Известно, что в случае больших задач ЛП проекционный метод может потребовать значительных временных затрат [65]. Потому мы разработали параллельную версию алгоритма 1, представленную в виде алгоритма 2. Параллельный алгоритм построен на основе модели параллельных вычислений BSF [66], ориентированной на кластерные вычислительные системы. Модель BSF использует схему распараллеливания «мастер–рабочие» и требует представление алгоритма в виде операций над списками с использованием функций высшего порядка *Map* и *Reduce*. В качестве второго параметра функции высшего порядка *Map* в алгоритме 2 используется список $\mathcal{L}_{map} = [1, \ldots, m]$, содержащий порядковые номера ограничений задачи ЛП (37), а в качестве первого параметра фигурирует параметризованная функция

$$\mathrm{F}_x : \mathcal{P} \to \mathbb{R}^n \times \mathbb{Z}_{\geqslant 0},$$

определенная следующим образом:

$$\begin{aligned} &\mathrm{F}_x(i) = (u_i, \sigma_i)\,; \\ &u_i = \begin{cases} \pi_i(x), & \text{если } \langle a_i, x\rangle > b_i; \\ \mathbf{0}, & \text{если } \langle a_i, x\rangle \leqslant b_i; \end{cases} \\ &\sigma_i = \begin{cases} 1, & \text{если } \langle a_i, x\rangle > b_i; \\ 0, & \text{если } \langle a_i, x\rangle \leqslant b_i. \end{cases} \end{aligned} \tag{100}$$

Алгоритм 1 Последовательное вычисление псевдопроекции $\rho_M(x)$

Require: $\hat{H}_i = \{x \in \mathbb{R}^n | \langle a_i, x\rangle \leqslant b_i\}$, $M = \bigcap_{i=1}^m \hat{H}_i$, $M \neq \emptyset$

function $\rho_M(x)$
  $k := 0$
  $x^{(0)} := x$
  repeat
    $\mathcal{J} := \emptyset$
    for $i = 1 \ldots m$ **do**
      if $\left\langle a_i, x^{(k)} \right\rangle > b_i$ **then**
        $\mathcal{J} := \mathcal{J} \cup \{i\}$
      end if
    end for
    if $\mathcal{J} = \emptyset$ **then**
      return $x^{(k)}$
    end if
    $S := 0$
    for all $i \in \mathcal{J}$ **do**
      $S := S + \left(\left\langle a_i, x^{(k)} \right\rangle - b_i\right) a_i / \left\|a_i\right\|^2$
    end for
    $x^{(k+1)} := x^{(k)} - S / \left|\mathcal{J}\right|$
    $k := k + 1$
  until $\left\|x^{(k)} - x^{(k-1)}\right\| < \epsilon$
  return $x^{(k)}$
end function

Таким образом, функция высшего порядка $Map\left(\mathrm{F}_x, \mathcal{L}_{map}\right)$ преобразует список номеров ограничений $\mathcal{L}_{map}$ в список пар (u_i, σ_i):

$$Map\left(\mathrm{F}_x, \mathcal{L}_{map}\right) = \left[\mathrm{F}_x(1), \ldots, \mathrm{F}_x(m)\right] = \left[(u_1, \sigma_1), \ldots, (u_m, \sigma_m)\right]. \tag{101}$$

Здесь u_i является ортогональной проекцией точки x на гиперплоскость H_i в том случае, когда $x \notin \hat{H}_i$, и нулевым вектором в противном; σ_i соответственно принимает значение 1 или 0. Обозначим $\mathcal{L}_{reduce} = [(u_1, \sigma_1), \ldots, (u_m, \sigma_m)]$. Определим бинарную ассоциативную операцию

$$\oplus : \mathbb{R}^n \times \mathbb{Z}_{\geqslant 0} \to \mathbb{R}^n \times \mathbb{Z}_{\geqslant 0},$$

являющуюся первым параметром функции высшего порядка *Reduce*:

$$\left(u', \sigma'\right) \oplus \left(u'', \sigma''\right) = \left(u' + u'', \sigma' + \sigma''\right). \tag{102}$$

Функция высшего порядка $Reduce\left(\oplus, \mathcal{L}_{reduce}\right)$ редуцирует список $\mathcal{L}_{reduce}$ к одной паре путем последовательного применения операции $\oplus$ ко всем элементам списка:

$$Reduce\left(\oplus, \mathcal{L}_{reduce}\right) = (u_1, \sigma_1) \oplus \ldots \oplus (u_m, \sigma_m) = (u, \sigma), \tag{103}$$

Алгоритм 2 Параллельное вычисление псевдопроекции $\rho_M(x)$

	мастер		l**-тый рабочий** $(l = 0, \ldots, L-1)$
1:	**input** $n, x^{(0)}$	1:	**input** n, m, A, b, c
2:		2:	$L :=$ NumberOfWorkers
3:	$k := 0$	3:	$\mathcal{L}_{map(l)} := [lm/L, \ldots, ((l+1)m/L) - 1]$
4:	**repeat**	4:	**repeat**
5:	**Bcast** $x^{(k)}$	5:	**RecvFromMaster** $x^{(k)}$
6:		6:	$\mathcal{L}_{reduce(l)} := Map(\mathrm{F}_{x^{(k)}}, \mathcal{L}_{map(l)})$
7:		7:	$(u_l, \sigma_l) := Reduce(\oplus, \mathcal{L}_{reduce(l)})$
8:	**Gather** $\mathcal{L}_{reduce}$	8:	**SendToMaster** (u_l, σ_l)
9:	$(u, \sigma) := Reduce(\oplus, \mathcal{L}_{reduce})$	9:	
10:	$x^{(k+1)} := u/\sigma$	10:	
11:	$k := k + 1$	11:	
12:	$exit := \left\|x^{(k)} - x^{(k-1)}\right\| < \epsilon$	12:	
13:	**Bcast** $exit$	13:	**RecvFromMaster** $exit$
14:	**until** $exit$	14:	**until** $exit$
15:	**output** $x^{(k)}$	15:	
16:	**stop**	16:	**stop**

где

$$u = \sum_{i=1}^{m} u_i; \tag{104}$$

$$\sigma = \sum_{i=1}^{m} \sigma_i. \tag{105}$$

Параллельная работа алгоритма 2 организована по схеме «мастер–рабочие» и включает в себя $L+1$ процесс: один процесс–мастер и L процессов–рабочих. Процесс–мастер осуществляет общее управление вычислениями, распределяет работу между процессами–рабочими, получает от них результаты и формирует итоговый результат. Для простоты будем предполагать, что количество ограничений m в задаче ЛП (37) кратно количеству рабочих L. На шаге 1 мастер вводит исходные данные: размерность пространства n и начальную точку $x^{(0)}$. На шаге 3 мастер присваивает счетчику итераций k значение 0. Шаги 4–14 реализуют основной цикл **repeat**–**until**, вычисляющий псевдопроекцию. На шаге 5 мастер рассылает текущее приближение $x^{(k)}$ всем рабочим. На шаге 8 он получает от рабочих частичные результаты, которые на шаге 9 редуцируются в пару (u, σ). Последняя используется на шаге 10 для вычисления следующего приближения $x^{(k+1)}$. На шаге 11 мастер увеличивает на единицу счетчик итераций k. На шаге 12 мастер проверяет условие завершения и присваивает результат проверки логической переменной $exit$. На шаге 13 мастер рассылает всем рабочим значение логической переменной $exit$. Если логическая переменная $exit$ принимает значение «истина», цикл **repeat**–**until** завершается на шаге 14. На шаге 15

мастер выводит последнее приближение $x^{(k)}$ в качестве результата псевдопроекции. Шаг 16 завершает работу процесса–мастера.

Все рабочие выполняют один и тот же код, но над различными данными. На шаге 1 l-тый рабочий вводит исходные данные задачи ЛП. Затем он формирует подсписок своих номеров ограничений для обработки (шаги 2–3). Для удобства программирования нумерация ограничений начинается с нуля. Подсписки различных рабочих не пересекаются, и их объединение дает полный список номеров ограничений:

$$\mathcal{L}_{map} = \mathcal{L}_{map(0)} +\!\!+ \ldots +\!\!+ \mathcal{L}_{map(L-1)}. \tag{106}$$

Символ $+\!\!+$ здесь обозначает операцию конкатенации списков. Цикл **repeat–until** рабочего соответствует циклу **repeat–until** мастера (шаги 4–14). На шаге 5 рабочий получает от мастера текущее приближение $x^{(k)}$. На шаге 6 рабочий вызывает функцию высшего порядка Map, которая, в свою очередь, применяет параметризованную функцию $\mathrm{F}_{x^{(k)}}$, определенную по формуле (100), ко всем элементам подсписка $\mathcal{L}_{map(l)}$, формируя на выходе подсписок пар $\mathcal{L}_{reduce(l)}$. Этот подсписок на шаге 7 редуцируется рабочим в единственную пару (u_l, σ_l) с помощью функции высшего порядка $Reduce$, которая последовательно применяет бинарную операцию $\oplus$, определенную по формуле (102), ко всем элементам подсписка $\mathcal{L}_{reduce(l)}$. На шаге 13 рабочий получает от мастера значение логической переменной $exit$. Если эта переменная принимает значение «истина», то рабочий процесс завершается. В противном случае продолжает выполняться цикл **repeat–until**. Операторы обмена **Bcast**, **Gather**, **RecvFromMaster** и **SendToMaster** обеспечивают неявную синхронизацию работы процесса–мастера и процессов–рабочих.

Выполним оценку границы масштабируемости описанного параллельного алгоритма, используя стоимостную метрику модели BSF [66]. Под границей масштабируемости параллельного алгоритма понимается максимальное число процессорных узлов, до которого наблюдается рост ускорения. Стоимостная метрика модели BSF включает в себя следующие параметры.

- m : длина списка $\mathcal{L}_{map}$;
- D : латентность (время, необходимое мастеру, чтобы послать одному рабочему сообщение длиной в один байт);
- t_c : время, необходимое мастеру, чтобы переслать одному рабочему текущее приближение $x^{(k)}$ и получить от него пару (u_l, σ_l) с учетом латентности;
- t_{Map} : время, требуемое одному рабочему, чтобы выполнить функцию высшего порядка Map для всех элементов списка $\mathcal{L}_{map}$;
- t_a : время, необходимое для выполнения бинарной операции $\oplus$, определяемой по формуле (102).

Согласно формуле (14) из [66], граница масштабируемости L_{max} параллельного алгоритма 2 может быть оценена следующим образом:

$$L_{max} = \frac{1}{2}\sqrt{\left(\frac{t_c}{t_a \ln 2}\right)^2 + \frac{t_{Map}}{t_a} + 4m} - \frac{t_c}{t_a \ln 2}. \tag{107}$$

Вычислим временные параметры в формуле (107). Введем следующие обозначения для одной итерации цикла **repeat–until** (шаги 4–14 алгоритма 2):

c_c : количество чисел, пересылаемых от мастера рабочему и обратно в ходе одной итерации;

c_F : количество арифметических операций и операций сравнения, необходимых для вычисления функции F_x, определяемой по формуле (100);

$c_\oplus$: количество арифметических операций и операций сравнения, необходимых для выполнения бинарной операции $\oplus$, определяемой по формуле (102).

На шаге 5 мастер посылает l-тому рабочему вектор размерности n. Затем на шаге 8 мастер получает от l-того рабочего пару, состоящую из вектора размерности n и одного вещественного числа. Кроме этого, на шаге 13 мастер посылает l-тому рабочему одно логическое значение. Последняя пересылка состоит в пересылке одного бита и равносильна одному добавлению латентности D, что будет сделано позже. Следовательно,

$$c_c = 2n + 1. \tag{108}$$

Принимая во внимание формулы (91), (100) и предполагая, что значения $\|a_i\|^2$ для всех $i = 1, \ldots, m$ вычислены заранее, получаем

$$c_F = 3n + 2. \tag{109}$$

Исходя из (102), для $c_\oplus$ справедлива следующая формула:

$$c_\oplus = 2n + 1. \tag{110}$$

Обозначим через τ_{op} время выполнения одной арифметической операции или операции сравнения. Обозначим через τ_{tr} время пересылки одного вещественного числа без учета латентности. Тогда на основе (108), (109) и (110) имеем

$$t_c = c_c\tau_{tr} + 3D = (2n + 1)\tau_{tr} + 3D; \tag{111}$$

$$t_{Map} = c_F m\tau_{op} = (3n + 2)m\tau_{op}; \tag{112}$$

$$t_a = c_\oplus\tau_{op} = (2n + 1)\tau_{op}. \tag{113}$$

Подставляя правые части этих формул в (107) и добавляя латентность D, получаем

$$L_{max} = \frac{1}{2}\sqrt{\left(\frac{(2n+1)\tau_{tr} + 3D}{(2n+1)\tau_{op}\ln 2}\right)^2 + \left(\frac{n+1}{2n+1} + 5\right)m} - \frac{(2n+1)\tau_{tr} + 3D}{(2n+1)\tau_{op}\ln 2},$$

где n — размерность пространства, m — количество ограничений. Для больших значений n и m отсюда вытекает следующая приближенная оценка:

$$L_{max} \approx O(\sqrt{m}). \tag{114}$$

Полученная оценка свидетельствует о том, что параллельный алгоритм 2 обладает слабой масштабируемостью[1].

[1]Если граница масштабируемости определяется формулой $L_{max} = O\left(m^\alpha\right)$, то мы полагаем, что параллельный алгоритм обладает сильной масштабируемостью при $\alpha \geqslant 1$, слабой масштабируемостью при $0 < \alpha < 1$, и масштабируемость отсутствует при $\alpha \leqslant 0$.

3.2. Стадия Quest

Стадия Quest играет роль предиктора и состоит из следующих шагов.

1. Найти допустимую точку $\tilde{x} \in M$.
2. Вычислить точку апекса z.
3. Построить начальное приближение $u^{(0)}$, являющееся псевдопроекцией точки апекса z на допустимый многогранник M.

Допустимая точка $\tilde{x}$ на шаге 1 может быть вычислена с помощью формулы

$$\tilde{x} = \begin{cases} \mathbf{0}, & \text{если } \mathbf{0} \in M; \\ \rho_M(\mathbf{0}), & \text{если } \mathbf{0} \notin M, \end{cases} \tag{115}$$

где $\rho_M(\cdot)$ — операция псевдопроектирования на допустимый многогранник M (см. определение 4).

Точка апекса z на шаге 2 может быть вычислена следующим образом:

$$z = \tilde{x} + \left(\eta + \max \left\{ \frac{b_i - \langle a_i, x' \rangle}{\langle a_i, e_c \rangle} \,\middle|\, i \in \mathcal{I} \right\} \right) e_c, \tag{116}$$

где $\mathcal{I}$ — множество индексов, для которых полупространство $\hat{H}_i$ является c-рецессивным; $\eta \in \mathbb{R}_{>0}$ — положительный параметр, определяющий удаление точки z от точки $\tilde{x}$. Следствие 1 гарантирует, что при любом $\eta > 0$ точка z, вычисленная по формуле (116), не принадлежит никакому рецессивному полупространству $\hat{H}_i$. Подобный выбор точки апекса z основывается на эвристике, согласно которой псевдопроекция такой точки будет находиться «не очень далеко» от точного решения задачи ЛП. Данная эвристика основана на утверждении 3, в котором говорится, что решение задачи ЛП (37) лежит на некоторой гиперплоскости H_i, ограничивающей рецессивное полупространство $\hat{H}_i$. При этом значение параметра η может существенно влиять на близость точки $\rho_M(z)$ к точному решению. Оптимальное значение η может быть получено путем нахождения максимума целевой функции с использованием метода последовательной дихотомии.

На шаге 3 вычисляется точка $u^{(0)}$ по формуле

$$u^{(0)} = \rho_M(z). \tag{117}$$

Эта точка служит начальным приближением на стадии Target. Многочисленные вычислительные эксперименты, выполненные нами на искусственных и реальных невырожденных задачах ЛП, показывают, что итерационный процесс вычисления псевдопроекции, стартуя с произвольной внешней точки, всегда сходится к точке на границе допустимого многогранника M. Однако, в настоящий момент у нас отсутствует строгое доказательство этого факта.

3.3. Стадия Target

Стадия Target играет в апекс-методе роль корректора и вычисляет последовательность точек

$$\left\{ u^{(0)}, u^{(1)}, \ldots, u^{(k)}, \ldots \right\}, \tag{118}$$

обладающую следующими свойствами:

$$u^{(k)} \in \Gamma_M; \tag{119}$$

Алгоритм 3 Стадия Target

Require: $\hat{H}_i = \{x \in \mathbb{R}^n | \langle a_i, x\rangle \leqslant b_i\}$, $M = \bigcap_{i=1}^m \hat{H}_i$, $M \neq \emptyset$

1: **input** $u^{(0)}$
2: $k := 0$
3: $v := u^{(k)} + \delta e_c$
4: $w := \rho_M(v)$
5: **while** $\left\langle c, w - u^{(k)}\right\rangle > \epsilon_f$ **do**
6: **assert** $\exists i \in \mathcal{I} : w, u^{(k)} \in H_i$ ▷ Если не выполняется, уменьшить δ
7: $d := w - u^{(k)}$
8: $\lambda' = \max\left\{\lambda \in \mathbb{R}_{>0} | \, u^{(k)} + \lambda d \in M\right\}$
9: $u^{(k+1)} := u^{(k)} + \lambda' d$
10: $k := k + 1$
11: $v := u^{(k)} + \delta e_c$
12: $w := \rho_M(v)$
13: **end while**
14: **output** $u^{(k)}$
15: **stop**

$$\left\langle c, u^{(k)}\right\rangle < \left\langle c, u^{(k+1)}\right\rangle; \tag{120}$$

$$\lim_{k\to\infty} \left\| u^{(k)} - \bar{x}\right\| = 0 \tag{121}$$

для всех $k \in \{0, 1, 2, \ldots\}$. Здесь Γ_M обозначает множество граничных точек допустимого многогранника M. Условие (119) означает, что все точки последовательности (118) лежат на границе допустимого многогранника M. Условие (120) говорит о том, что значение целевой функции в каждой точке последовательности (118) больше, чем в предыдущей. Согласно условию (121) последовательность (118) сходится к точному решению задачи ЛП (37).

Реализация стадии Target приведена в виде алгоритма 3. Дадим краткие комментарии по шагам алгоритма 3. На шаге 1 осуществляется ввод начального приближения $u^{(0)}$, полученного на стадии Quest. На шаге 2 счетчику итераций k присваивается значение 0. На шаге 3 вычисляется внешняя точка v как сумма векторов δe_c и $u^{(k)}$. Здесь e_c обозначает единичный вектор, сонаправленный с вектором c. На шаге 4 вычисляется точка w, являющаяся псевдопроекцией точки v на допустимый многогранник M. Шаги 5–13 реализуют основной цикл стадии Target, проиллюстрированный на рис. 1. Этот цикл выполняется, пока справедливо условие

$$\left\langle c, w - u^{(k)}\right\rangle > \epsilon_f. \tag{122}$$

Здесь ϵ_f — малый положительный параметр. На шаге 6 проверяется требование, в соответствии с которым точки w и $u_{(k)}$ должны лежать на некоторой гиперплоскости H_i, ограничивающей рецессивное полупространство $\hat{H}_i$. Это необходимо для того, чтобы перемещение от точки $u^{(k)}$ к точке $u^{(k+1)}$ происходило по поверхности многогранника M, а не через его внутреннюю часть. Если это требование не выполняется, необходимо уменьшить параметр δ. На шаге 7 вычисляется вектор d, задающий направление перемещения. На шаге 8 вычисляется

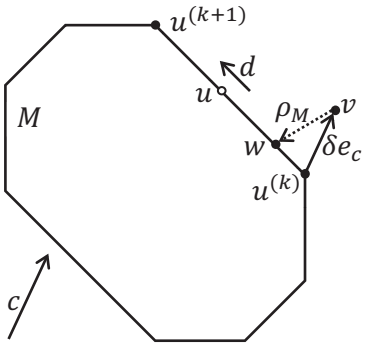

Рис. 1. Итерация основного цикла стадии Target

максимальное положительное число λ', для которого точка $\left(u^{(k)} + \lambda' d\right)$ принадлежит многограннику M. На шаге 9 вычисляется следующее приближение $u^{(k+1)}$. На шаге 10 счетчик итераций k увеличивается на 1. На шагах 11 и 12 вычисляются новая внешняя точка v и ее псевдопроекция w, используемые на следующей итерации основного цикла. После выхода из основного цикла на шаге 14 точка $u^{(k)}$ выводится в качестве приближенного решения задачи ЛП (37).

Следующее утверждение гарантирует сходимость алгоритма 3.

Утверждение 6. Пусть допустимый многогранник M задачи ЛП (37) является непустым ограниченным множеством. Тогда последовательность $\left\{u^{(k)}\right\}$, генерируемая алгоритмом 3, завершается через конечное число итераций $K \geqslant 0$ в некоторой допустимой точке, причем

$$\left\langle c, u^{(0)} \right\rangle < \left\langle c, u^{(1)} \right\rangle < \left\langle c, u^{(2)} \right\rangle < \ldots < \left\langle c, u^{(K)} \right\rangle. \tag{123}$$

Доказательство. Случай $K = 0$ является тривиальным. Пусть $K > 0$, либо $K = \infty$. Сначала покажем, что для любого $k < K$ выполняется следующее неравенство:

$$\left\langle c, u^{(k)} \right\rangle < \left\langle c, u^{(k+1)} \right\rangle. \tag{124}$$

Действительно, из (122) следует, что

$$\left\langle c, u^{(k)} \right\rangle < \langle c, w \rangle. \tag{125}$$

Принимая во внимание шаг 7 алгоритма 3, это означает, что

$$d \neq \mathbf{0}. \tag{126}$$

В соответствии с шагами 8, 9 имеем

$$u^{(k+1)} = u^{(k)} + \lambda' d, \tag{127}$$

где $\lambda' > 0$. Принимая во внимание неравенство (122) и шаг 7 алгоритма 3, отсюда следует

$$\begin{aligned} &\left\langle c, u^{(k+1)} \right\rangle = \left\langle c, u^{(k)} + \lambda' d \right\rangle = \left\langle c, u^{(k)} + \lambda' \left(w - u^{(k)} \right) \right\rangle = \\ &= \left\langle c, u^{(k)} \right\rangle + \lambda' \left\langle c, w - u^{(k)} \right\rangle > \left\langle c, u^{(k)} \right\rangle. \end{aligned}$$

Теперь покажем, что $K < \infty$. Предположим противное, то есть алгоритм 3 генерирует бесконечную последовательность точек. В таком случае мы получаем бесконечную монотонно возрастающую числовую последовательность

$$\left\langle c, u^{(0)} \right\rangle < \left\langle c, u^{(1)} \right\rangle < \left\langle c, u^{(2)} \right\rangle < \dots \tag{128}$$

Поскольку допустимый многогранник M является ограниченным множеством, последовательность (128) ограничена сверху. Согласно теореме Вейерштрасса монотонно возрастающая ограниченная числовая последовательность имеет конечный предел, равный ее супремуму. Это означает, что существует $K' \in \mathbb{N}$ такой, что

$$\forall k > K' : \left\langle c, u^{(k+1)} \right\rangle - \left\langle c, u^{(k)} \right\rangle < \epsilon_f. \tag{129}$$

Отсюда следует

$$\forall k > K' : \langle c, w \rangle - \left\langle c, u^{(k)} \right\rangle < \epsilon_f, \tag{130}$$

что равносильно

$$\forall k > K' : \left\langle c, w - u^{(k)} \right\rangle < \epsilon_f. \tag{131}$$

Получили противоречие с условием (122) выполнения цикла, используемом на шаге 5 алгоритма 3. *Утверждение доказано.* □

Покажем, что последовательность $\left\{u^{(k)}\right\}$, генерируемая алгоритмом 3, сходится к точному решению задачи ЛП (37) при $\epsilon_f \to 0$. Для этого заметим, что при $\delta \to 0$ псевдопроекция сводится к метрической проекции. Следуя [38], дадим определение метрической проекции.

Определение 5. Пусть Q является замкнутым выпуклым множеством в $\mathbb{R}^n$, и $Q \neq \emptyset$. Метрическая проекция $P_Q(x)$ точки $x \in \mathbb{R}^n$ на множество Q определяется формулой

$$P_Q(x) = \arg\min \left\{ \|x - q\| \,|q \in Q \right\}. \tag{132}$$

Следующее утверждение имеет место.

Утверждение 7. Последовательность $\left\{u^{(k)}\right\}$, генерируемая алгоритмом 3 с метрической проекцией $P_M(\cdot)$ вместо псевдопроекции $\rho_M(\cdot)$, завершается через конечное число итераций $K \geqslant 0$ в некоторой допустимой точке, причем

$$\left\langle c, u^{(0)} \right\rangle < \left\langle c, u^{(1)} \right\rangle < \left\langle c, u^{(2)} \right\rangle < \dots < \left\langle c, u^{(K)} \right\rangle. \tag{133}$$

Доказательство. Данное утверждение доказывается по той же схеме, что и утверждение 6. □

Следующее утверждение доказывает сходимость алгоритма 3 к точному решению задачи ЛП (37) для случая метрической проекции.

Утверждение 8. При замене псевдопроекции $\rho_M(\cdot)$ метрической проекцией $P_M(\cdot)$ алгоритм 3 завершается через конечное число итераций в точке $\bar{x}$, являющейся точным решением задачи ЛП (37).

Доказательство. Обозначим через $\bar{u}$ конечную точку последовательности $\left\{u^{(k)}\right\}$, генерируемой алгоритмом 3 с использованием метрической проекции $P_M(\cdot)$. Такая точка существует в силу утверждения 7. Предположим противное, то есть $\bar{u} \neq \bar{x}$. Это равносильно

$$\langle c, \bar{u} \rangle < \langle c, \bar{x} \rangle . \tag{134}$$

Обозначим с помощью $S_\delta(v)$ открытый n-мерный шар радиуса δ с центром в точке v, где

$$v = \bar{u} + \delta e_c. \tag{135}$$

В силу (134) имеем

$$S_\delta(v) \cap M \neq \emptyset. \tag{136}$$

Положим

$$w = \arg\min \{ \|x - v\| \, | x \in S_\delta(v) \cap M \} . \tag{137}$$

Последнее эквивалентно

$$w = P_M(v). \tag{138}$$

Легко видеть, что справедливо неравенство

$$\langle c, w \rangle > \langle c, \bar{u} \rangle . \tag{139}$$

Сопоставляя формулу (135) с шагом 11 алгоритма 3, формулу (138) с шагом 12 (где псевдопроекция заменена на метрическую проекцию), и формулу (139) с условием на шаге 5, мы видим, что $\bar{u}$ не может быть конечной точкой последовательности $\left\{u^{(k)}\right\}$, генерируемой алгоритмом 3. Получили противоречие. *Утверждение доказано.* □

На практике заменить псевдопроекцию $\rho_M(v)$ в алгоритме 3 на метрическую проекцию $P_M(v)$ не представляется возможным, так как неизвестен алгоритм вычисления метрической проекции на выпуклый замкнутый многогранник в общем случае. Таким образом, утверждение 8 в строгом смысле не доказывает сходимость алгоритма 3 к точному решению задачи ЛП (37), хотя на практике такая сходимость наблюдалась нами во всех случаях.

4. Программная реализация и вычислительные эксперименты

Мы реализовали параллельную версию апекс-метода на языке C++ с использованием программного BSF-каркаса [67], базирующегося на модели параллельных вычислений BSF [66]. BSF-каркас инкапсулирует все аспекты, связанные с распараллеливанием программы на основе библиотеки MPI. Исходные коды апекс-метода свободно доступны в репозитории GitHub по адресу `https://github.com/leonid-sokolinsky/Apex-method`. С помощью этой программы мы исследовали масштабируемость апекс-метода. Масштабные вычислительные эксперименты проводились на вычислительном кластере «Торнадо ЮУрГУ» [68], характеристики которого представлены в табл. 1. В качестве тестов мы использовали искусственные задачи, полученные с помощью генератора случайных задач линейного программирования FRaGenLP [69]. Верификация решений, выдаваемых апекс-методом, осуществлялась программой VaLiPro [70]. Была выполнена серия вычислительных экспериментов, в которой для задач ЛП различной размерности исследовались ускорение и параллельная эффективность в зависимости от количества используемых рабочих узлов. Результаты этих экспериментов представлены на рис. 2. В данном контексте ускорение $\alpha(L)$ опре-

Таблица 1. Характеристики кластера «Торнадо ЮУрГУ»

Параметр	Значение
Количество процессорных узлов	480
Процессоры	Intel Xeon X5680 (6 cores, 3.33 GHz)
Число процессоров на узел	2
Память на узел	24 GB DDR3
Соединительная сеть	InfiniBand QDR (40 Gbit/s)
Операционная система	Linux CentOS

делялось как отношение времени $T(1)$ решения задачи на конфигурации с узлом-мастером и единственным узлом-рабочим ко времени $T(L)$ решения той же задачи на конфигурации с узлом-мастером и L узлами-рабочими:

$$\alpha(L) = \frac{T(1)}{T(L)}. \tag{140}$$

Параллельная эффективность $\epsilon(L)$ вычислялась как отношение ускорения $\alpha(L)$ к числу L используемых узлов-рабочих:

$$\epsilon(L) = \frac{\alpha(L)}{L}. \tag{141}$$

Вычисления проводились для следующих размерностей: 5 000, 7 500 и 10 000. Число ограничений соответственно составило 10 002, 15 002 и 20 002.

Эксперименты показали, что граница масштабируемости параллельной реализации апекс-метода существенно зависит от размера задачи. Для $n = 5\,000$ граница масштабируемости составила приблизительно 55 рабочих узлов. Для задачи размерности $n = 7\,500$ эта граница увеличилась до 80 узлов, а для задачи размерности $n = 10\,000$ она оказалась близкой к 100 узлам. Дальнейшее увеличение размерности задачи приводило к ошибке компилятора «недостаточно памяти». Необходимо отметить, что вычисления проводились с двойной точностью, при которой число с плавающей точкой занимает в оперативной памяти 64 бита. Попытка использовать одинарную точность, требующую 32 бита для хранения числа с плавающей точкой, оказалась неудачной, так как при этом апекс-метод переставал сходиться к точному решению задачи ЛП.

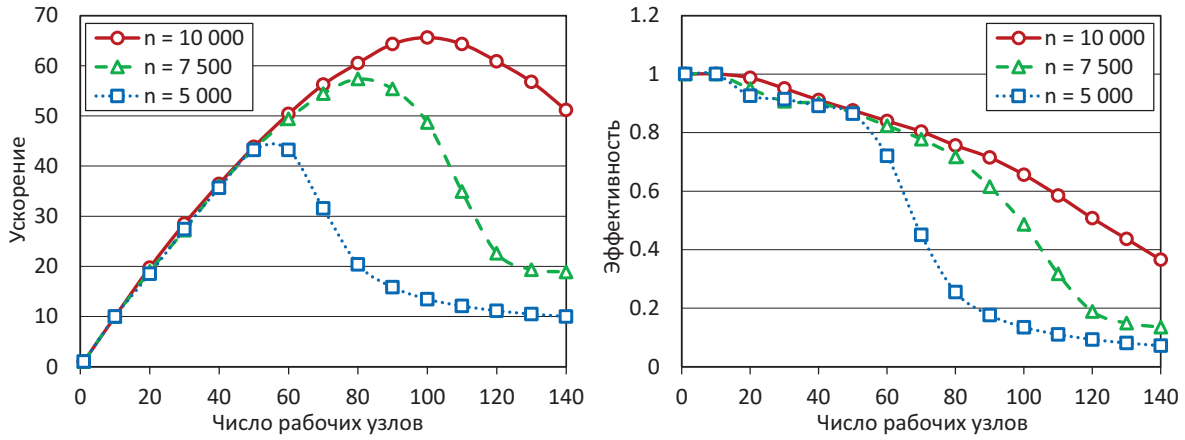

Рис. 2. Ускорение и параллельная эффективность апекс-метода

Параллельная эффективность также продемонстрировала существенную зависимость от размера задачи ЛП. При $n = 5\,000$ эффективность показала падение ниже 50% уже на 70 рабочих узлах. Для $n = 7\,500$ и $n = 10\,000$ падение на 50% наблюдалось на 110 и 130 рабочих узлах соответственно.

Кроме этого, эксперименты показали, что параметр η в формуле (116), используемой на стадии Quest для вычисления точки апекса z, оказывает незначительное влияние на общее время решения задачи ЛП в случае, когда этот параметр принимает большие значения (более 100 000). Если точка апекса располагается недостаточно далеко от допустимого многогранника, то ее псевдопроекция может оказаться на одной из его граней. Если же точка апекса располагается далеко от допустимого многогранника (в экспериментах использовалось значение $\eta = 20000n$), то ее псевдопроекция всегда оказывается в одной из его вершин. Также стоит отметить, что для искусственных задач, сгенерированных программой FRaGenLP, все точки последовательности $\left\{u^{(k)}\right\}$ оказывались на пути, близком к оптимальному[2].

Проведенные вычислительные эксперименты на искусственных задачах показали, что более 99% времени апекс-метод тратил на вычисление псевдопроекций (шаг 18 алгоритма 3). При этом вычисление одного приближения $u^{(k)}$ для задачи размерности $n = 10\,000$ на 100 рабочих узлах занимало 44 минуты.

Мы также протестировали апекс-метод на задачах из репозитория Netlib-LP [71], доступного по адресу `https://netlib.org/lp/data`. Набор задач линейной оптимизации Netlib-LP включает в себя множество реальных приложений, таких как оценка лесных ресурсов, задачи нефтепереработки, проектирование закрылков самолетов, модели пилотирования, планирование работы аудиторского персонала, расчет мостовых ферм, планирование расписаний авиакомпаний, расчет моделей промышленного производства и распределения ресурсов, восстановление изображений и задачи многосекторального экономического планирования. Netlib-LP содержит задачи ЛП размером от 32 переменных и 27 ограничений до 15 695 переменных и 16 675 ограничений [72]. Точные решения (оптимальные значения целевых функций) для всех задач были заимствованы из работы [73]. Результаты представлены в табл. 2. Эксперименты показали, что относительная ошибка грубого приближения, вычисляемого на стадии Quest, не превосходила 0.2 для всех задач, кроме *adlittle*, *blend*, и *fit1d*. Относительная ошибка уточненного приближения, получаемого на стадии Target, оказалась менее 10^{-3}, за исключением задач *kb2* и *sc105*, для которых ошибка составила 0.035 и 0.007 соответственно. Время решения указанных задач варьировалось от нескольких секунд для *afiro* до десятков часов для *blend*. Одним из главных параметров, влияющих на скорость сходимости апекс-метода, был параметр ϵ, используемый на шаге 12 параллельного алгоритма 2, вычисляющего псевдопроекцию. Прогоны всех задач доступны на GitHub по адресу `https://github.com/leonid-sokolinsky/Apex-method/tree/master/Runs`.

5. Обсуждение полученных результатов

В этом разделе мы обсудим научную и практическую значимость апекс-метода, его сильные и слабые стороны, и дадим ответы на следующие вопросы.

1. В чем состоит научная значимость полученных результатов?
2. В чем заключается практическое значение апекс-метода?

[2]Под оптимальным путем понимается путь движения по поверхности допустимого многогранника в направлении максимального, в данном случае, увеличения значения целевой функции.

Таблица 2. Применение апекс-метода для решения задач из Netlib-LP

№	Задача из Netlib-LP		Стадия Quest		Стадия Target	
	Наименование	Точное решение	Грубое приближение	Ошибка	Уточненное приближение	Ошибка
1	adlittle	2.25494963E5	3.67140280E5	6.28E-1	2.2571324E5	9.68E-4
2	afiro	-4.64753142E2	-4.55961488E2	1.89E-2	-4.6475310E2	8.61E-9
3	blend	-3.08121498E1	-3.60232513E0	8.83E-1	-3.0811018E1	3.19E-5
4	fit1d	-9.14637809E3	-3.49931014E3	6.17E-1	-9.1463386E3	8.77E-7
5	kb2	-1.74990012E3	-1.39603193E3	2.02E-1	-1.6879152E3	3.54E-2
6	recipe	-2.66616000E2	-2.66107349E2	1.91E-3	-2.6660404E2	2.23E-5
7	sc50a	-6.45750770E1	-5.58016335E1	1.36E-1	-6.4568167E1	1.06E-4
8	sc50b	-7.00000000E1	-6.92167246E1	1.12E-2	-6.9990792E1	1.32E-4
9	sc105	-5.22020612E1	-4.28785710E1	1.79E-1	-5.1837995E1	6.97E-3
10	share2b	-4.15732240E2	-4.28792528E2	3.14E-2	-4.1572001E2	2.40E-5

3. На сколько мы можем быть уверены, что апекс-метод всегда сходится к точному решению задачи ЛП?
4. Как мы можем ускорить сходимость апекс-метода в целом и выполнение операции псевдопроектирования в частности?

Основной научный вклад этой работы заключается в том, что разработан апекс-метод, впервые позволяющий построить на поверхности допустимого многогранника близкий к оптимальному путь от начальной точки до точки решения задачи ЛП. Под оптимальным путем мы понимаем путь движения по поверхности многогранника в направлении максимального увеличения или уменьшения значения целевой функции в зависимости от того, ее максимум или минимум необходимо найти.

Практическая значимость апекс-метода состоит в том, что он открывает возможность использования искусственных нейронных сетей прямого распространения, включая сверточные нейронные сети, для решения многомерных задач ЛП. В недавней работе [48] был предложен оригинальный метод визуализации n-мерных задач ЛП. Этот метод строит образ допустимого многогранника M в виде матрицы I размерности $(n-1)$ на основе техники растеризации. В качестве луча зрения используется вектор, противоположно направленный вектору градиента целевой функции. Каждый пиксель представляется в матрице I вещественным числом, пропорциональным значению целевой функции в соответствующей точке на поверхности допустимого многогранника M. Подобные образы подаются на вход нейронной сети прямого распространения для нахождения оптимального пути к решению задачи ЛП. Более точно, нейронная сеть прямого распространения непосредственно вычисляет вектор d в алгоритме 3, делая ненужным ресурсоемкие вычисления псевдопроекций. Главным преимуществом такого подхода является то, что нейронная сеть прямого распространения работает в режиме реального времени, актуальном для задач робототехники. В настоящее время нам не известны другие методы решения задач ЛП, работающие в режиме реального времени. Однако применение нейронных сетей прямого распространения для решения задач ЛП предполагает подготовку обучающих наборов данных. Апекс-метод впервые предоставляет возможность конструировать такие обучающие наборы.

В утверждении 6 говорится, что алгоритм 3 сходится за конечное число итераций к некоторой точке на поверхности допустимого многогранника M, однако вопрос о том, будет ли эта точка решением задачи ЛП, остается открытым. Согласно утверждению 8 ответ на этот вопрос оказывается положительным, если в алгоритме 3 заменить псевдопроекцию на метрическую проекцию. Однако не существует методов построения метрической проекции для произвольного выпуклого замкнутого многогранника. Поэтому мы вынуждены использовать псевдопроекцию. Многочисленные эксперименты показывают, что апекс-метод всегда сходится к решению задачи ЛП, однако этот факт нуждается в формальном доказательстве. Мы планируем получить такое доказательство в рамках наших дальнейших исследований.

Основным недостатком апекс-метода является его медленная сходимость к решению задачи ЛП. Задача ЛП, которая занимает несколько секунд для нахождения оптимального решения с использованием одного из стандартных решателей, может потребовать нескольких часов для нахождения решения с помощью апекс-метода. Вычислительные эксперименты показывают, что более 99% времени, затрачиваемого апекс-методом на решение задачи ЛП, приходится на вычисление псевдопроекций. Поэтому вопрос ускорения процесса вычисления псевдопроекций является актуальным. В апекс-методе псевдопроекции вычисляются с помощью алгоритма 1, принадлежащего к семейству проекционных методов, рассмотренных в разделе 1. Известно [74], что в случае, когда выпуклое замкнутое множество является многогранником $M \neq \emptyset$, методы проекционного типа имеют низкую линейную скорость сходимости:

$$\left\| x^{(k+1)} - \rho_M \left(x^{(0)} \right) \right\| \leqslant C q^k, \tag{142}$$

где $0 < C < \infty$ — некоторая константа, а $q \in (0, 1)$ — параметр, зависящий от углов между гиперплоскостями, соответствующими граням многогранника M. Это означает, что расстояние между соседними приближениями с каждой итерацией уменьшается в геометрической прогрессии со знаменателем, меньшим единицы. Для малых углов скорость сходимости может падать до значений, близких к нулю. Это фундаментальное ограничение методов проекционного типа не может быть преодолено. Однако, мы можем уменьшить количество гиперплоскостей, вовлекаемых в вычисление псевдопроекции. В соответствии с утверждением 3 решение задачи ЛП (37) находится на границе некоторого рецессивного полупространства. Следовательно, при вычислении псевдопроекции с помощью алгоритма 1, нам достаточно использовать только гиперплоскости, ограничивающие рецессивные полупространства. Это уменьшает количество ограничений в среднем в два раза. Другой способ сократить время вычисления псевдопроекций — распараллелить алгоритм 1, как это было сделано в алгоритме 2. Однако в этом случае степень параллелизма будет ограничена теоретической оценкой (114).

Заключение

В статье предложен новый масштабируемый итерационный метод линейного программирования, получивший название «апекс-метод». Ключевой особенностью этого метода является построение максимального пути на поверхности допустимого многогранника от начальной токи к решению задачи линейного программирования. Под максимальным путем понимается путь движения по поверхности допустимого многогранника в направлении максимального увеличения значения целевой функции. Практическая значимость предложенного метода состоит в том, что он открывает возможность применения искусственных

нейронных сетей прямого распространения для решения многомерных задач линейного программирования.

В работе описан оригинальный теоретический базис, лежащий в основе апекс-метода. Рассмотрены полупространства, порождаемые ограничениями задачи линейного программирования. Пересечение этих полупространств образует замкнутый выпуклый многогранник M, называемый допустимым. Указанные полупространства делятся на две группы, в зависимости от градиента c линейной целевой функции: доминантные и рецессивные. Получено достаточное и необходимое условие для того, чтобы полупространство было рецессивным.

Доказано, что решение задачи линейного программирования всегда лежит на границе некоторого рецессивного полупространства. Получена формула вычисления точки апекса, которая не принадлежит ни одному рецессивному полупространству. Точка апекса используется для получения начального приближения на поверхности допустимого многогранника M. Для построения оптимального пути к решению задачи линейного программирования апекс-метод использует параллельный алгоритм построения псевдопроекции, являющейся обобщением метрической проекции. Для параллельного алгоритма построения псевдопроекции получена аналитическая оценка границы его масштабируемости на кластерной вычислительной системе. Эта граница не превышает $O\left(\sqrt{m}\right)$ процессорных узлов, где m — количество ограничений задачи линейного программирования. Описан алгоритм, строящий на границе допустимого многогранника оптимальный путь от начального приближения до точки решения задачи линейного программирования. Доказана сходимость этого алгоритма.

Параллельная версия апекс-метода реализована на языке C++ с использованием программного BSF-каркаса, основанного на модели параллельных вычислений BSF. Проведены эксперименты по исследованию масштабируемости апекс-метода на кластерной вычислительной системе. Вычислительные эксперименты показали, что для задачи линейного программирования с 10 000 переменными и 20 002 ограничениями граница масштабируемости не превышает 100 процессорных узлов. В то же время эксперименты показали, что более 99% времени, затрачиваемого на решение задачи линейного программирования апекс-методом, приходилось на вычисление псевдопроекций.

В дополнение, апекс-метод был протестирован на 10 задачах из репозитория Netlib-LP. Относительная ошибка на этих задачах составила от $3.5{\cdot}10^{-3}$ до $8.6{\cdot}10^{-9}$. Время вычислений варьировалось от нескольких секунд до нескольких десятков часов. Точность вычисления псевдопроекции оказалась основным параметром, влияющим на скорость сходимости апекс-метода.

В качестве направлений дальнейших исследований выделим следующие. Мы планируем разработать новый, более эффективный метод вычисления псевдопроекций на допустимый многогранник. Основная идея состоит в сокращении количества полупространств, используемых в рамках одной итерации. В то же время количество оставшихся в рассмотрении полупространств должно быть достаточным для эффективного распараллеливания. Мы рассчитываем, что новый метод превзойдет алгоритм 2 по скорости сходимости. Также мы собираемся доказать, что новый метод сходится к точке, лежащей на границе допустимого многогранника. Кроме этого мы планируем исследовать полезность использования в апекс-методе техники линейной супериоризации, предложенной в работе [64].

Обозначения

$\mathbb{R}^n$	вещественное евклидово пространство
$\|\cdot\|$	евклидова норма
$\langle\cdot,\cdot\rangle$	скалярное произведение двух векторов
$[\cdot\,,\cdot]$	конкатенация двух векторов
$f(x)$	линейная целевая функция
c	градиент целевой функции $f(x)$
e_c	единичный вектор, сонаправленный с вектором c
$\bar{x}$	решение задачи ЛП
M	допустимый многогранник
Γ_M	множество граничных точек допустимого многогранника M
a_i	i-тая строка матрицы A
$\hat{H}_i$	полупространство, определяемое формулой $\langle a_i, x\rangle \leqslant b_i$
H_i	гиперплоскость, определяемая формулой $\langle a_i, x\rangle = b_i$
$\mathcal{P}$	множество индексов строк матрицы A
$\mathcal{I}$	множество индексов, для которых полупространство $\hat{H}_i$ является рецессивным
$\pi_i(\cdot)$	ортогональная проекция на гиперплоскость H_i
$\rho_M(\cdot)$	псевдопроекция на допустимый многогранник M
$P_M(\cdot)$	метрическая проекция на допустимый многогранник M

Исследование выполнено при финансовой поддержке РНФ (проект №23-21-00356).

Литература

1. Соколинская И.М., Соколинский Л.Б. Об одном итерационном методе решения задач линейного программирования на кластерных вычислительных системах // Вычислительные методы и программирование. 2020. Т. 21, № 4. С. 329–340. DOI: 10.26089/NumMet.v21r328.

2. Branke J. Optimization in Dynamic Environments // Evolutionary Optimization in Dynamic Environments. Genetic Algorithms and Evolutionary Computation, vol. 3. Boston, MA: Springer, 2002. P. 13–29. DOI: 10.1007/978-1-4615-0911-0_2.

3. Brogaard J., Hendershott T., Riordan R. High-Frequency Trading and Price Discovery // Review of Financial Studies. 2014. Vol. 27, no. 8. P. 2267–2306. DOI: 10.1093/rfs/hhu032.

4. Deng S., Huang X., Wang J., *et al.* A Decision Support System for Trading in Apple Futures Market Using Predictions Fusion // IEEE Access. 2021. Vol. 9. P. 1271–1285. DOI: 10.1109/ACCESS.2020.3047138.

5. Seregin G. Lecture notes on regularity theory for the Navier-Stokes equations. Singapore: World Scientific Publishing Company, 2014. 268 p. DOI: 10.1142/9314.

6. Demin D.A. Synthesis of optimal control of technological processes based on a multialternative parametric description of the final state // Eastern-European Journal of Enterprise Technologies. 2017. Vol. 3, 4(87). P. 51–63. DOI: 10.15587/1729-4061.2017.105294.

7. Kazarinov L.S., Shnayder D.A., Kolesnikova O.V. Heat load control in steam boilers // 2017 International Conference on Industrial Engineering, Applications and Manufacturing, ICIEAM 2017 - Proceedings. IEEE, 2017. DOI: 10.1109/ICIEAM.2017.8076177.

8. Zagoskina E.V., Barbasova T.A., Shnaider D.A. Intelligent Control System of Blast-furnace Melting Efficiency // SIBIRCON 2019 - International Multi-Conference on Engineering, Computer and Information Sciences, Proceedings. IEEE, 2019. P. 710–713. DOI: 10.1109/SIBIRCON48586.2019.8958221.

9. Fleming J., Yan X., Allison C., *et al.* Real-time predictive eco-driving assistance considering road geometry and long-range radar measurements // IET Intelligent Transport Systems. 2021. Vol. 15, no. 4. P. 573–583. DOI: 10.1049/ITR2.12047.

10. Scholl M., Minnerup K., Reiter C., *et al.* Optimization of a thermal management system for battery electric vehicles // 14th International Conference on Ecological Vehicles and Renewable Energies, EVER 2019. IEEE, 2019. DOI: 10.1109/EVER.2019.8813657.

11. Meisel S. Dynamic Vehicle Routing // Anticipatory Optimization for Dynamic Decision Making. Operations Research/Computer Science Interfaces Series, vol. 51. New York, NY: Springer, 2011. P. 77–96. DOI: 10.1007/978-1-4614-0505-4_6.

12. Cheng A.M.K. Real-Time Scheduling and Schedulability Analysis // Real-Time Systems: Scheduling, Analysis, and Verification. John Wiley, Sons, 2002. P. 41–85. DOI: 10.1002/0471224626.CH3.

13. Kopetz H. Real-Time Scheduling // Real-Time Systems. Real-Time Systems Series. Boston, MA: Springer, 2011. P. 239–258. DOI: 10.1007/978-1-4419-8237-7_10.

14. Dantzig G.B. Linear programming and extensions. Princeton, N.J.: Princeton university press, 1998. 656 p.

15. Hall J., McKinnon K. Hyper-sparsity in the revised simplex method and how to exploit it // Computational Optimization and Applications. 2005. Vol. 32, no. 3. P. 259–283. DOI: 10.1007/s10589-005-4802-0.

16. Klee V., Minty G. How good is the simplex algorithm? // Inequalities - III. Proceedings of the Third Symposium on Inequalities Held at the University of California, Los Angeles, Sept. 1-9, 1969 / ed. by O. Shisha. New York-London: Academic Press, 1972. P. 159–175.

17. Jeroslow R. The simplex algorithm with the pivot rule of maximizing criterion improvement // Discrete Mathematics. 1973. Vol. 4, no. 4. P. 367–377. DOI: 10.1016/0012-365X(73)90171-4.

18. Zadeh N. A bad network problem for the simplex method and other minimum cost flow algorithms // Mathematical Programming. 1973. Vol. 5, no. 1. P. 255–266. DOI: 10.1007/BF01580132.

19. Bartels R., Stoer J., Zenger C. A Realization of the Simplex Method Based on Triangular Decompositions // Handbook for Automatic Computation. Volume II: Linear Algebra. Berlin, Heidelberg: Springer, 1971. P. 152–190. DOI: 10.1007/978-3-642-86940-2_11.

20. Tolla P. A Survey of Some Linear Programming Methods // Concepts of Combinatorial Optimization / ed. by V.T. Paschos. 2nd ed. Hoboken, NJ, USA: John Wiley, Sons, 2014. Chap. 7. P. 157–188. DOI: 10.1002/9781119005216.ch7.

21. Hall J. Towards a practical parallelisation of the simplex method // Computational Management Science. 2010. Vol. 7, no. 2. P. 139–170. DOI: 10.1007/s10287-008-0080-5.

22. Mamalis B., Pantziou G. Advances in the Parallelization of the Simplex Method // Algorithms, Probability, Networks, and Games. Lecture Notes in Computer Science, vol. 9295 / ed. by C. Zaroliagis, G. Pantziou, S. Kontogiannis. Cham: Springer, 2015. P. 281–307. DOI: 10.1007/978-3-319-24024-4_17.

23. Lubin M., Hall J.A.J., Petra C.G., Anitescu M. Parallel distributed-memory simplex for large-scale stochastic LP problems // Computational Optimization and Applications. 2013. Vol. 55, no. 3. P. 571–596. DOI: 10.1007/s10589-013-9542-y.

24. Хачиян Л. Полиномиальные алгоритмы в линейном программировании // Журнал вычислительной математики и математической физики. 1980. Т. 20, № 1. С. 51–68.

25. Шор Н.З. Метод отсечения с растяжением пространства для решения задач выпуклого программирования // Кибернетика. 1977. № 1. С. 94–95.

26. Юдин Д.Б., Немировский А.С. Информационная сложность и эффективные методы решения выпуклых экстремальных задач // Экономика и математические методы. 1976. № 2. С. 357–369.

27. Karmarkar N. A new polynomial-time algorithm for linear programming // Combinatorica. 1984. Vol. 4, no. 4. P. 373–395. DOI: 10.1007/BF02579150.

28. Дикин И.И. Итеративное решение задач линейного и квадратичного программирования // Доклады Академии наук СССР. 1967. Т. 174, № 4. С. 747–748. URL: https://www.mathnet.ru/rus/dan33112.

29. Gondzio J. Interior point methods 25 years later // European Journal of Operational Research. 2012. Vol. 218, no. 3. P. 587–601. DOI: 10.1016/j.ejor.2011.09.017.

30. Зоркальцев В.И., Мокрый И.В. Алгоритмы внутренних точек в линейной оптимизации // Сибирский журнал индустриальной математики. 2018. Т. 21, 1 (73). С. 11–20.

31. Fathi-Hafshejani S., Mansouri H., Reza Peyghami M., Chen S. Primal–dual interior-point method for linear optimization based on a kernel function with trigonometric growth term // Optimization. 2018. Vol. 67, no. 10. P. 1605–1630. DOI: 10.1080/02331934.2018.1482297.

32. Asadi S., Mansouri H. A Mehrotra type predictor-corrector interior-point algorithm for linear programming // Numerical Algebra, Control and Optimization. 2019. Vol. 9, no. 2. P. 147–156. DOI: 10.3934/naco.2019011.

33. Yuan Y. Implementation tricks of interior-point methods for large-scale linear programs // Proc. SPIE, vol. 8285. International Conference on Graphic and Image Processing (ICGIP 2011). International Society for Optics, Photonics, 2011. DOI: 10.1117/12.913019.

34. Kheirfam B., Haghighi M. A full-Newton step infeasible interior-point method for linear optimization based on a trigonometric kernel function // Optimization. 2016. Vol. 65, no. 4. P. 841–857. DOI: 10.1080/02331934.2015.1080255.

35. Xu Y., Zhang L., Zhang J. A full-modified-newton step infeasible interior-point algorithm for linear optimization // Journal of Industrial and Management Optimization. 2016. Vol. 12, no. 1. P. 103–116. DOI: 10.3934/jimo.2016.12.103.

36. Roos C., Terlaky T., Vial J.-P. Interior Point Methods for Linear Optimization. New York: Springer, 2005. 500 p. DOI: 10.1007/b100325.

37. Sokolinskaya I. Parallel Method of Pseudoprojection for Linear Inequalities // Parallel Computational Technologies. PCT 2018. Communications in Computer and Information Science, vol. 910 / ed. by L. Sokolinsky, M. Zymbler. Cham: Springer, 2018. P. 216–231. DOI: 10.1007/978-3-319-99673-8_16.

38. Васин В.В., Ерёмин И.И. Операторы и итерационные процессы фейеровского типа. Теория и приложения. Екатеринбург: УрО РАН, 2005. 211 с.

39. Gondzio J., Grothey A. Direct Solution of Linear Systems of Size 109 Arising in Optimization with Interior Point Methods // Parallel Processing and Applied Mathematics. PPAM 2005. Lecture Notes in Computer Science, vol. 3911. 3911 LNCS / ed. by R. Wyrzykowski, J. Dongarra, N. Meyer, J. Wasniewski. Berlin, Heidelberg: Springer, 2006. P. 513–525. DOI: 10.1007/11752578_62.

40. Prieto A., Prieto B., Ortigosa E.M., *et al.* Neural networks: An overview of early research, current frameworks and new challenges // Neurocomputing. 2016. Vol. 214. P. 242–268. DOI: 10.1016/j.neucom.2016.06.014.

41. Raina R., Madhavan A., Ng A.Y. Large-scale deep unsupervised learning using graphics processors // Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09). New York, NY, USA: ACM Press, 2009. P. 873–880. DOI: 10.1145/1553374.1553486.

42. Tank D.W., Hopfield J.J. Simple 'neural' optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit // IEEE transactions on circuits and systems. 1986. Vol. CAS-33, no. 5. P. 533–541. DOI: 10.1109/TCS.1986.1085953.

43. Kennedy M.P., Chua L.O. Unifying the Tank and Hopfield Linear Programming Circuit and the Canonical Nonlinear Programming Circuit of Chua and Lin // IEEE Transactions on Circuits and Systems. 1987. Vol. 34, no. 2. P. 210–214. DOI: 10.1109/TCS.1987.1086095.

44. Rodriguez-Vazquez A., Dominguez-Castro R., Rueda A., *et al.* Nonlinear Switched-Capacitor "Neural" Networks for Optimization Problems // IEEE Transactions on Circuits and Systems. 1990. Vol. 37, no. 3. P. 384–398. DOI: 10.1109/31.52732.

45. Zak S.H., Upatising V. Solving Linear Programming Problems with Neural Networks: A Comparative Study // IEEE Transactions on Neural Networks. 1995. Vol. 6, no. 1. P. 94–104. DOI: 10.1109/72.363446.

46. Malek A., Yari A. Primal–dual solution for the linear programming problems using neural networks // Applied Mathematics and Computation. 2005. Vol. 167, no. 1. P. 198–211. DOI: 10.1016/J.AMC.2004.06.081.

47. Liu X., Zhou M. A one-layer recurrent neural network for non-smooth convex optimization subject to linear inequality constraints // Chaos, Solitons and Fractals. 2016. Vol. 87. P. 39–46. DOI: 10.1016/j.chaos.2016.03.009.

48. Ольховский Н.А., Соколинский Л.Б. Визуальное представление многомерных задач линейного программирования // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2022. Т. 11, № 1. С. 31–56. DOI: 10.14529/cmse220103.

49. LeCun Y., Bengio Y., Hinton G. Deep learning // Nature. 2015. Vol. 521, no. 7553. P. 436–444. DOI: 10.1038/nature14539.

50. Lachhwani K. Application of Neural Network Models for Mathematical Programming Problems: A State of Art Review // Archives of Computational Methods in Engineering. 2020. Vol. 27. P. 171–182. DOI: 10.1007/s11831-018-09309-5.

51. Поляк Б.Т. Рандомизированные алгоритмы решения выпуклых неравенств // Стохастическая оптимизация в информатике / под ред. О.Н. Граничин. СПб.: Издательство С.-Петербургского университета, 2005. С. 123–127. URL: https://www.math.spbu.ru/user/gran/sb1/polyak.pdf.

52. Kaczmarz S. Angenherte Auflsung von Systemen linearer Gleichungen // Bulletin International de l'Acadmie Polonaise des Sciences et des Lettres. Classe des Sciences Mathmatiques et Naturelles. Srie A, Sciences Mathmatiques. 1937. Vol. 35. P. 355–357.

53. Kaczmarz S. Approximate solution of systems of linear equations // International Journal of Control. 1993. Vol. 57, no. 6. P. 1269–1271. DOI: 10.1080/00207179308934446.

54. Cimmino G. Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari // La Ricerca Scientifica, XVI, Series II, Anno IX, 1. 1938. P. 326–333.

55. Gastinel N. Linear Numerical Analysis. New York: Academic Press, 1971. ix+341 p.

56. Agmon S. The relaxation method for linear inequalities // Canadian Journal of Mathematics. 1954. Vol. 6. P. 382–392. DOI: 10.4153/CJM-1954-037-2.

57. Motzkin T.S., Schoenberg I.J. The relaxation method for linear inequalities // Canadian Journal of Mathematics. 1954. Vol. 6. P. 393–404. DOI: 10.4153/CJM-1954-038-x.

58. Censor Y., Elfving T. New methods for linear inequalities // Linear Algebra and its Applications. 1982. Vol. 42. P. 199–211. DOI: 10.1016/0024-3795(82)90149-5.

59. De Pierro A.R., Iusem A.N. A simultaneous projections method for linear inequalities // Linear Algebra and its Applications. 1985. Vol. 64. P. 243–253. DOI: 10.1016/0024-3795(85)90280-0.

60. Соколинская И., Соколинский Л. Исследование масштабируемости алгоритма Чиммино для решения систем линейных неравенств на кластерных вычислительных системах // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2019. Т. 8, № 1. С. 20–35. DOI: 10.14529/cmse190102.

61. Соколинский Л.Б., Соколинская И.М. Параллельный алгоритм решения нестационарных систем линейных неравенств // Параллельные вычислительные технологии – XIV международная конференция, ПаВТ'2020, г. Пермь, 31 марта–2 апреля 2020 г. Короткие статьи и описания плакатов. Челябинск: Издательский центр ЮУрГУ, 2020. С. 275–286. DOI: 10.14529/pct2020.

62. Ерёмин И.И., Попов Л.Д. Фейеровские процессы в теории и практике: обзор последних результатов // Известия вузов. Математика. 2009. № 1. С. 44–65. URL: https://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=ivm&paperid=1253.

63. Nurminski E.A. Single-projection procedure for linear optimization // Journal of Global Optimization. 2016. Vol. 66, no. 1. P. 95–110. DOI: 10.1007/S10898-015-0337-9.

64. Censor Y. Can linear superiorization be useful for linear optimization problems? // Inverse Problems. 2017. Vol. 33, no. 4. P. 044006. DOI: 10.1088/1361-6420/33/4/044006.

65. Gould N.I. How good are projection methods for convex feasibility problems? // Computational Optimization and Applications. 2008. Vol. 40, no. 1. P. 1–12. DOI: 10.1007/S10589-007-9073-5.

66. Sokolinsky L.B. BSF: A parallel computation model for scalability estimation of iterative numerical algorithms on cluster computing systems // Journal of Parallel and Distributed Computing. 2021. Vol. 149. P. 193–206. DOI: 10.1016/j.jpdc.2020.12.009.

67. Sokolinsky L.B. BSF-skeleton: A Template for Parallelization of Iterative Numerical Algorithms on Cluster Computing Systems // MethodsX. 2021. Vol. 8. Article number 101437. DOI: 10.1016/j.mex.2021.101437.

68. Dolganina N., Ivanova E., Bilenko R., Rekachinsky A. HPC Resources of South Ural State University // Parallel Computational Technologies. PCT 2022. Communications in Computer and Information Science, vol. 1618 / ed. by L. Sokolinsky, M. Zymbler. Cham: Springer, 2022. P. 43–55. DOI: 10.1007/978-3-031-11623-0_4.

69. Соколинский Л.Б., Соколинская И.М. О генерации случайных задач линейного программирования на кластерных вычислительных системах // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 2. С. 38–52. DOI: 10.14529/cmse210103.

70. Соколинский Л.Б., Соколинская И.М. О валидации решений задач линейного программирования на кластерных вычислительных системах // Вычислительные методы и программирование. 2021. Т. 22, № 4. С. 252–261. DOI: 10.26089/NUMMET.V22R416.

71. Gay D.M. Electronic mail distribution of linear programming test problems // Mathematical Programming Society COAL Bulletin. 1985. Vol. 13. P. 10–12.

72. Keil C., Jansson C. Computational experience with rigorous error bounds for the Netlib linear programming library // Reliable Computing. 2006. Vol. 12, no. 4. P. 303–321. DOI: 10.1007/S11155-006-9004-7/METRICS.

73. Koch T. The final NETLIB-LP results // Operations Research Letters. 2004. Vol. 32, no. 2. P. 138–142. DOI: 10.1016/S0167-6377(03)00094-4.

74. Deutsch F., Hundal H. The rate of convergence for the cyclic projections algorithm I: Angles between convex sets // Journal of Approximation Theory. 2006. Vol. 142, no. 1. P. 36–55. DOI: 10.1016/J.JAT.2006.02.005.

Соколинский Леонид Борисович, д.ф.-м.н., профессор, заведующий кафедрой системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Соколинская Ирина Михайловна, к.ф.-м.н., доцент кафедры математического обеспечения информационных технологий, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse230201

ON NEW VERSION OF THE APEX METHOD FOR SOLVING LINEAR PROGRAMMING PROBLEMS

© 2023 L.B. Sokolinsky, I.M. Sokolinskaya

South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)

E-mail: leonid.sokolinsky@susu.ru, irina.sokolinskaya@susu.ru

Received: 07.04.2023

The article presents a new scalable iterative method for linear programming, called the apex method. The key feature of this method is constructing a path close to optimal on the surface of the feasible region from a certain starting point to the exact solution of the linear programming problem. The optimal path refers to a path of minimum length according to the Euclidean metric. The apex method is based on the predictor-corrector framework and proceeds in two stages: Quest (predictor) and Target (corrector). The Quest stage calculates a rough initial approximation of the linear programming problem. The Target stage refines the initial approximation with a given precision. The main operation used in the apex method is an operation that calculates the pseudoprojection, which is a generalization of the metric projection to a convex closed set. This operation is used both in the Quest stage and in the Target stage. A parallel algorithm using a Fejér mapping to compute the pseudoprojection is presented. An analytical estimation of the parallelism degree of this algorithm is obtained. Also, an algorithm implementing the Target stage is given. The convergence of this algorithm is proven. The results of applying the apex method for solving various linear programming problems are presented.

Keywords: linear programming, apex method, iterative method, projection-type method, Fejér mapping, parallel algorithm, scalability evaluation.

FOR CITATION

Sokolinsky L.B., Sokolinskaya I.M. On New Version of the Apex Method for Solving Linear Programming Problems. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2023. Vol. 12, no. 2. P. 5–46. (in Russian) DOI: 10.14529/cmse230201.

References

1. Sokolinsky L.B., Sokolinskaya I.M. Scalable Method for Linear Optimization of Industrial Processes. Proceedings - 2020 Global Smart Industry Conference, GloSIC 2020. IEEE, 2020. 20–26. Article number 9267854. DOI: 10.1109/GloSIC50886.2020.9267854.
2. Branke J. Optimization in Dynamic Environments. Evolutionary Optimization in Dynamic Environments. Genetic Algorithms and Evolutionary Computation, vol. 3. Boston, MA: Springer, 2002. P. 13–29. DOI: 10.1007/978-1-4615-0911-0_2.
3. Brogaard J., Hendershott T., Riordan R. High-Frequency Trading and Price Discovery. Review of Financial Studies. 2014. Vol. 27, no. 8. P. 2267–2306. DOI: 10.1093/rfs/hhu032.
4. Deng S., Huang X., Wang J., *et al.* A Decision Support System for Trading in Apple Futures Market Using Predictions Fusion. IEEE Access. 2021. Vol. 9. P. 1271–1285. DOI: 10.1109/ACCESS.2020.3047138.

5. Seregin G. Lecture notes on regularity theory for the Navier-Stokes equations. Singapore: World Scientific Publishing Company, 2014. 268 p. DOI: 10.1142/9314.

6. Demin D.A. Synthesis of optimal control of technological processes based on a multialternative parametric description of the final state. Eastern-European Journal of Enterprise Technologies. 2017. Vol. 3, 4(87). P. 51–63. DOI: 10.15587/1729-4061.2017.105294.

7. Kazarinov L.S., Shnayder D.A., Kolesnikova O.V. Heat load control in steam boilers. 2017 International Conference on Industrial Engineering, Applications and Manufacturing, ICIEAM 2017 - Proceedings. IEEE, 2017. DOI: 10.1109/ICIEAM.2017.8076177.

8. Zagoskina E.V., Barbasova T.A., Shnaider D.A. Intelligent Control System of Blast-furnace Melting Efficiency. SIBIRCON 2019 - International Multi-Conference on Engineering, Computer and Information Sciences, Proceedings. IEEE, 2019. P. 710–713. DOI: 10.1109/SIBIRCON48586.2019.8958221.

9. Fleming J., Yan X., Allison C., *et al.* Real-time predictive eco-driving assistance considering road geometry and long-range radar measurements. IET Intelligent Transport Systems. 2021. Vol. 15, no. 4. P. 573–583. DOI: 10.1049/ITR2.12047.

10. Scholl M., Minnerup K., Reiter C., *et al.* Optimization of a thermal management system for battery electric vehicles. 14th International Conference on Ecological Vehicles and Renewable Energies, EVER 2019. IEEE, 2019. DOI: 10.1109/EVER.2019.8813657.

11. Meisel S. Dynamic Vehicle Routing. Anticipatory Optimization for Dynamic Decision Making. Operations Research/Computer Science Interfaces Series, vol. 51. New York, NY: Springer, 2011. P. 77–96. DOI: 10.1007/978-1-4614-0505-4_6.

12. Cheng A.M.K. Real-Time Scheduling and Schedulability Analysis. Real-Time Systems: Scheduling, Analysis, and Verification. John Wiley, Sons, 2002. P. 41–85. DOI: 10.1002/0471224626.CH3.

13. Kopetz H. Real-Time Scheduling. Real-Time Systems. Real-Time Systems Series. Boston, MA: Springer, 2011. P. 239–258. DOI: 10.1007/978-1-4419-8237-7_10.

14. Dantzig G.B. Linear programming and extensions. Princeton, N.J.: Princeton university press, 1998. 656 p.

15. Hall J., McKinnon K. Hyper-sparsity in the revised simplex method and how to exploit it. Computational Optimization and Applications. 2005. Vol. 32, no. 3. P. 259–283. DOI: 10.1007/s10589-005-4802-0.

16. Klee V., Minty G. How good is the simplex algorithm?. Inequalities - III. Proceedings of the Third Symposium on Inequalities Held at the University of California, Los Angeles, Sept. 1-9, 1969 / ed. by O. Shisha. New York-London: Academic Press, 1972. P. 159–175.

17. Jeroslow R. The simplex algorithm with the pivot rule of maximizing criterion improvement. Discrete Mathematics. 1973. Vol. 4, no. 4. P. 367–377. DOI: 10.1016/0012-365X(73)90171-4.

18. Zadeh N. A bad network problem for the simplex method and other minimum cost flow algorithms. Mathematical Programming. 1973. Vol. 5, no. 1. P. 255–266. DOI: 10.1007/BF01580132.

19. Bartels R., Stoer J., Zenger C. A Realization of the Simplex Method Based on Triangular Decompositions. Handbook for Automatic Computation. Volume II: Linear Algebra. Berlin, Heidelberg: Springer, 1971. P. 152–190. DOI: 10.1007/978-3-642-86940-2_11.

20. Tolla P. A Survey of Some Linear Programming Methods. Concepts of Combinatorial Optimization / ed. by V.T. Paschos. 2nd ed. Hoboken, NJ, USA: John Wiley, Sons, 2014. Chap. 7. P. 157–188. DOI: 10.1002/9781119005216.ch7.

21. Hall J. Towards a practical parallelisation of the simplex method. Computational Management Science. 2010. Vol. 7, no. 2. P. 139–170. DOI: 10.1007/s10287-008-0080-5.

22. Mamalis B., Pantziou G. Advances in the Parallelization of the Simplex Method. Algorithms, Probability, Networks, and Games. Lecture Notes in Computer Science, vol. 9295 / ed. by C. Zaroliagis, G. Pantziou, S. Kontogiannis. Cham: Springer, 2015. P. 281–307. DOI: 10.1007/978-3-319-24024-4_17.

23. Lubin M., Hall J.A.J., Petra C.G., Anitescu M. Parallel distributed-memory simplex for large-scale stochastic LP problems. Computational Optimization and Applications. 2013. Vol. 55, no. 3. P. 571–596. DOI: 10.1007/s10589-013-9542-y.

24. Khachiyan L. Polynomial algorithms in linear programming. USSR Computational Mathematics and Mathematical Physics. 1980. Vol. 20, no. 1. P. 53–72. DOI: 10.1016/0041-5553(80)90061-0.

25. Shor N.Z. Cut-off method with space extension in convex programming problems. Cybernetics and Systems Analysis. 1977. Vol. 13, no. 1. P. 94–96. DOI: 10.1007/BF01071394.

26. Yudin D., Nemirovsky A. Information complexity and efficient methods for solving convex extremal problems. Economics and mathematical methods (Ekonomika i matematicheskie metody). 1976. No. 2. P. 357–369. (in Russian).

27. Karmarkar N. A new polynomial-time algorithm for linear programming. Combinatorica. 1984. Vol. 4, no. 4. P. 373–395. DOI: 10.1007/BF02579150.

28. Dikin I. Iterative solution of problems of linear and quadratic programming. Soviet Mathematics. Doklady. 1967. Vol. 8. P. 674–675.

29. Gondzio J. Interior point methods 25 years later. European Journal of Operational Research. 2012. Vol. 218, no. 3. P. 587–601. DOI: 10.1016/j.ejor.2011.09.017.

30. Zorkaltsev V., Mokryi I. Interior point algorithms in linear optimization. Journal of applied and industrial mathematics. 2018. Vol. 12, no. 1. P. 191–199. DOI: 10.1134/S1990478918010179.

31. Fathi-Hafshejani S., Mansouri H., Reza Peyghami M., Chen S. Primal–dual interior-point method for linear optimization based on a kernel function with trigonometric growth term. Optimization. 2018. Vol. 67, no. 10. P. 1605–1630. DOI: 10.1080/02331934.2018.1482297.

32. Asadi S., Mansouri H. A Mehrotra type predictor-corrector interior-point algorithm for linear programming. Numerical Algebra, Control and Optimization. 2019. Vol. 9, no. 2. P. 147–156. DOI: 10.3934/naco.2019011.

33. Yuan Y. Implementation tricks of interior-point methods for large-scale linear programs. Proc. SPIE, vol. 8285. International Conference on Graphic and Image Processing (ICGIP 2011). International Society for Optics, Photonics, 2011. DOI: 10.1117/12.913019.

34. Kheirfam B., Haghighi M. A full-Newton step infeasible interior-point method for linear optimization based on a trigonometric kernel function. Optimization. 2016. Vol. 65, no. 4. P. 841–857. DOI: 10.1080/02331934.2015.1080255.

35. Xu Y., Zhang L., Zhang J. A full-modified-newton step infeasible interior-point algorithm for linear optimization. Journal of Industrial and Management Optimization. 2016. Vol. 12, no. 1. P. 103–116. DOI: 10.3934/jimo.2016.12.103.

36. Roos C., Terlaky T., Vial J.-P. Interior Point Methods for Linear Optimization. New York: Springer, 2005. 500 p. DOI: 10.1007/b100325.

37. Sokolinskaya I. Parallel Method of Pseudoprojection for Linear Inequalities. Parallel Computational Technologies. PCT 2018. Communications in Computer and Information Science, vol. 910 / ed. by L. Sokolinsky, M. Zymbler. Cham: Springer, 2018. P. 216–231. DOI: 10.1007/978-3-319-99673-8_16.

38. Vasin V.V., Eremin I.I. Operators and Iterative Processes of Fejér Type. Theory and Applications. Berlin, New York: Walter de Gruyter, 2009. 155 p. Inverse and III-Posed Problems Series. DOI: 10.1515/9783110218190.

39. Gondzio J., Grothey A. Direct Solution of Linear Systems of Size 109 Arising in Optimization with Interior Point Methods. Parallel Processing and Applied Mathematics. PPAM 2005. Lecture Notes in Computer Science, vol. 3911. 3911 LNCS / ed. by R. Wyrzykowski, J. Dongarra, N. Meyer, J. Wasniewski. Berlin, Heidelberg: Springer, 2006. P. 513–525. DOI: 10.1007/11752578_62.

40. Prieto A., Prieto B., Ortigosa E.M., *et al.* Neural networks: An overview of early research, current frameworks and new challenges. Neurocomputing. 2016. Vol. 214. P. 242–268. DOI: 10.1016/j.neucom.2016.06.014.

41. Raina R., Madhavan A., Ng A.Y. Large-scale deep unsupervised learning using graphics processors. Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09). New York, NY, USA: ACM Press, 2009. P. 873–880. DOI: 10.1145/1553374.1553486.

42. Tank D.W., Hopfield J.J. Simple 'neural' optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit. IEEE transactions on circuits and systems. 1986. Vol. CAS-33, no. 5. P. 533–541. DOI: 10.1109/TCS.1986.1085953.

43. Kennedy M.P., Chua L.O. Unifying the Tank and Hopfield Linear Programming Circuit and the Canonical Nonlinear Programming Circuit of Chua and Lin. IEEE Transactions on Circuits and Systems. 1987. Vol. 34, no. 2. P. 210–214. DOI: 10.1109/TCS.1987.1086095.

44. Rodriguez-Vazquez A., Dominguez-Castro R., Rueda A., *et al.* Nonlinear Switched-Capacitor "Neural" Networks for Optimization Problems. IEEE Transactions on Circuits and Systems. 1990. Vol. 37, no. 3. P. 384–398. DOI: 10.1109/31.52732.

45. Zak S.H., Upatising V. Solving Linear Programming Problems with Neural Networks: A Comparative Study. IEEE Transactions on Neural Networks. 1995. Vol. 6, no. 1. P. 94–104. DOI: 10.1109/72.363446.

46. Malek A., Yari A. Primal–dual solution for the linear programming problems using neural networks. Applied Mathematics and Computation. 2005. Vol. 167, no. 1. P. 198–211. DOI: 10.1016/J.AMC.2004.06.081.

47. Liu X., Zhou M. A one-layer recurrent neural network for non-smooth convex optimization subject to linear inequality constraints. Chaos, Solitons and Fractals. 2016. Vol. 87. P. 39–46. DOI: 10.1016/j.chaos.2016.03.009.

48. Olkhovsky N., Sokolinsky L. Visualizing Multidimensional Linear Programming Problems. Parallel Computational Technologies. PCT 2022. Communications in Computer and Information Science, vol. 1618 / ed. by L. Sokolinsky, M. Zymbler. Cham: Springer, 2022. P. 172–196. DOI: 10.1007/978-3-031-11623-0_13.

49. LeCun Y., Bengio Y., Hinton G. Deep learning. Nature. 2015. Vol. 521, no. 7553. P. 436–444. DOI: 10.1038/nature14539.

50. Lachhwani K. Application of Neural Network Models for Mathematical Programming Problems: A State of Art Review. Archives of Computational Methods in Engineering. 2020. Vol. 27. P. 171–182. DOI: 10.1007/s11831-018-09309-5.

51. Polyak B.T. Random Algorithms for Solving Convex Inequalities. Studies in Computational Mathematics. Vol. 8 / ed. by C. Brezinski, L. Wuytack. Amsterdam, London, New York, Oxford, Paris, Shannon, Tokyo: Elsevier, 2001. P. 409–422. DOI: 10.1016/S1570-579X(01)80024-0.

52. Kaczmarz S. Angenherte Auflsung von Systemen linearer Gleichungen. Bulletin International de l'Acadmie Polonaise des Sciences et des Lettres. Classe des Sciences Mathmatiques et Naturelles. Srie A, Sciences Mathmatiques. 1937. Vol. 35. P. 355–357.

53. Kaczmarz S. Approximate solution of systems of linear equations. International Journal of Control. 1993. Vol. 57, no. 6. P. 1269–1271. DOI: 10.1080/00207179308934446.

54. Cimmino G. Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari. La Ricerca Scientifica, XVI, Series II, Anno IX, 1. 1938. P. 326–333.

55. Gastinel N. Linear Numerical Analysis. New York: Academic Press, 1971. ix+341 p.

56. Agmon S. The relaxation method for linear inequalities. Canadian Journal of Mathematics. 1954. Vol. 6. P. 382–392. DOI: 10.4153/CJM-1954-037-2.

57. Motzkin T.S., Schoenberg I.J. The relaxation method for linear inequalities. Canadian Journal of Mathematics. 1954. Vol. 6. P. 393–404. DOI: 10.4153/CJM-1954-038-x.

58. Censor Y., Elfving T. New methods for linear inequalities. Linear Algebra and its Applications. 1982. Vol. 42. P. 199–211. DOI: 10.1016/0024-3795(82)90149-5.

59. De Pierro A.R., Iusem A.N. A simultaneous projections method for linear inequalities. Linear Algebra and its Applications. 1985. Vol. 64. P. 243–253. DOI: 10.1016/0024-3795(85)90280-0.

60. Sokolinskaya I.M., Sokolinsky L.B. Scalability Evaluation of Cimmino Algorithm for Solving Linear Inequality Systems on Multiprocessors with Distributed Memory. Supercomputing Frontiers and Innovations. 2018. Vol. 5, no. 2. P. 11–22. DOI: 10.14529/jsfi180202.

61. Sokolinsky L.B., Sokolinskaya I.M. Scalable parallel algorithm for solving non-stationary systems of linear inequalities. Lobachevskii Journal of Mathematics. 2020. Vol. 41, no. 8. P. 1571–1580. DOI: 10.1134/S1995080220080181.

62. Eremin I.I., Popov L.D. Fejér processes in theory and practice: Recent results. Russian Mathematics. 2009. Vol. 53, no. 1. P. 36–55. DOI: 10.3103/S1066369X09010022.

63. Nurminski E.A. Single-projection procedure for linear optimization. Journal of Global Optimization. 2016. Vol. 66, no. 1. P. 95–110. DOI: 10.1007/S10898-015-0337-9.

64. Censor Y. Can linear superiorization be useful for linear optimization problems?. Inverse Problems. 2017. Vol. 33, no. 4. P. 044006. DOI: 10.1088/1361-6420/33/4/044006.

65. Gould N.I. How good are projection methods for convex feasibility problems?. Computational Optimization and Applications. 2008. Vol. 40, no. 1. P. 1–12. DOI: 10.1007/S10589-007-9073-5.

66. Sokolinsky L.B. BSF: A parallel computation model for scalability estimation of iterative numerical algorithms on cluster computing systems. Journal of Parallel and Distributed Computing. 2021. Vol. 149. P. 193–206. DOI: 10.1016/j.jpdc.2020.12.009.

67. Sokolinsky L.B. BSF-skeleton: A Template for Parallelization of Iterative Numerical Algorithms on Cluster Computing Systems. MethodsX. 2021. Vol. 8. Article number 101437. DOI: 10.1016/j.mex.2021.101437.

68. Dolganina N., Ivanova E., Bilenko R., Rekachinsky A. HPC Resources of South Ural State University. Parallel Computational Technologies. PCT 2022. Communications in Computer and Information Science, vol. 1618 / ed. by L. Sokolinsky, M. Zymbler. Cham: Springer, 2022. P. 43–55. DOI: 10.1007/978-3-031-11623-0_4.

69. Sokolinsky L.B., Sokolinskaya I.M. FRaGenLP: A Generator of Random Linear Programming Problems for Cluster Computing Systems. Parallel Computational Technologies. PCT 2021. Communications in Computer and Information Science, vol. 1437 / ed. by L. Sokolinsky, M. Zymbler. Cham: Springer, 2021. P. 164–177. DOI: 10.1007/978-3-030-81691-9_12.

70. Sokolinsky L.B., Sokolinskaya I.M. VaLiPro: Linear Programming Validator for Cluster Computing Systems. Supercomputing Frontiers and Innovations. 2021. Vol. 8, no. 3. P. 51–61. DOI: 10.14529/jsfi210303.

71. Gay D.M. Electronic mail distribution of linear programming test problems. Mathematical Programming Society COAL Bulletin. 1985. Vol. 13. P. 10–12.

72. Keil C., Jansson C. Computational experience with rigorous error bounds for the Netlib linear programming library. Reliable Computing. 2006. Vol. 12, no. 4. P. 303–321. DOI: 10.1007/S11155-006-9004-7/METRICS.

73. Koch T. The final NETLIB-LP results. Operations Research Letters. 2004. Vol. 32, no. 2. P. 138–142. DOI: 10.1016/S0167-6377(03)00094-4.

74. Deutsch F., Hundal H. The rate of convergence for the cyclic projections algorithm I: Angles between convex sets. Journal of Approximation Theory. 2006. Vol. 142, no. 1. P. 36–55. DOI: 10.1016/J.JAT.2006.02.005.