2023, **24** (4), 408–429. doi 10.26089/NumMet.v24r428

doi 10.26089/NumMet.v24r428 УДК 519.852

О новом методе линейного программирования

Н. А. Ольховский
Южно-Уральский государственный университет (национальный исследовательский университет), Челябинск, Российская Федерация
ORCID: 0009-0008-9078-4799, e-mail: olkhovskiina@susu.ru

Л. Б. Соколинский
Южно-Уральский государственный университет (национальный исследовательский университет), Челябинск, Российская Федерация
ORCID: 0000-0001-9997-3918, e-mail: leonid.sokolinsky@susu.ru

Аннотация: В статье представлен новый итерационный метод линейного программирования, получивший название “метод поверхностного движения”. Данный метод строит на поверхности многогранника, ограничивающего допустимую область задачи линейного программирования, путь от начальной граничной точки до точки, в которой достигается оптимальное значение целевой функции. Вектор движения строится в направлении максимального увеличения/уменьшения значения целевой функции. Представлено формальное описание алгоритма, реализующего метод поверхностного движения. Доказана теорема сходимости. Описанный метод предполагает использование глубокой нейронной сети прямого распространения для определения направления движения по граням допустимого многогранника. Для этого строится многомерный локальный образ задачи линейного программирования в точке текущего приближения, который подается на вход нейронной сети. Множество размеченных прецедентов, необходимое для обучения нейронной сети, может быть получено с помощью апекс-метода.

Ключевые слова: линейное программирование, метод поверхностного движения, итерационный метод, теорема сходимости, глубокая нейронная сеть.

Благодарности: Исследование выполнено при финансовой поддержке РНФ (проект № 23–21–00356).

Для цитирования: Ольховский Н.А., Соколинский Л.Б. О новом методе линейного программирования // Вычислительные методы и программирование. 2023. **24**, № 4. 408–429. doi 10.26089/NumMet.v24r428.

A new method of linear programming

Nikolay A. Olkhovsky
South Ural State University (National Research University), Chelyabinsk, Russia
ORCID: 0009-0008-9078-4799, e-mail: olkhovskiina@susu.ru

Leonid B. Sokolinsky
South Ural State University (National Research University), Chelyabinsk, Russia
ORCID: 0000-0001-9997-3918, e-mail: leonid.sokolinsky@susu.ru

Abstract: The article presents a new iterative method of linear programming, called the surface movement method. This method builds a path from the initial boundary point to the point at which the optimal value of the objective function is achieved on the surface of a polytope that restricts the feasible region of linear programming problem. The movement vector defines the direction of the maximum increase/decrease in the value of the objective function. A formal description of the

2023, **24** (4), 408–429. doi 10.26089/NumMet.v24r428

algorithm implementing the surface movement method is presented. The convergence theorem is proved. The described method involves the use of a feed forward deep neural network to determine the direction of movement along the edges of a feasible polytope. To do this, a multidimensional local image of the linear programming problem is constructed at the point of the current approximation, which is fed to the input of the neural network. The set of labeled precedents necessary for training a neural network can be obtained using the apex method.

Keywords: linear programming, surface motion method, iterative method, convergence theorem, deep neural network.

Acknowledgements: The research was supported by RSF (project No. 23–21–00356).

For citation: N.A. Olkhovsky and L.B. Sokolinsky, "A new method of linear programming using neural networks," Numerical Methods and Programming. **24** (4), 408–429 (2023). doi 10.26089/NumMet.v24r428.

1. Введение. Быстрое развитие технологий обработки и хранения больших данных [1] привело к возникновению оптимизационных математических моделей в виде задач линейного программирования (ЛП) большой размерности [2]. Особый интерес представляют нестационарные задачи ЛП, связанные с оптимизацией нестационарных процессов [3]. В нестационарной задаче ЛП ограничения и целевая функция могут меняться динамически в ходе ее решения [4]. Следующие оптимизационные задачи могут быть сведены к нестационарным задачам ЛП: выбор оптимальных стратегий в роботрейдинге [5, 6], оптимальное управление летательными аппаратами [7], оптимизация технологических процессов [8–10], логистические и транспортные задачи [11–13], планирование и управление производством продукции [14]. Отдельно можно упомянуть оптимизационные задачи, которые должны решаться в режиме реального времени [15]. В качестве примеров можно привести управление химическим производством, управление системой многоточечного впрыска топлива в ДВС, управление сотовыми сетями, автопилотирование, системы самонаведения ракет.

Простейший подход к решению нестационарных задач оптимизации заключается в том, что всякое изменение исходных данных воспринимается как отдельная новая задача [3]. Такой подход может быть приемлемым, когда изменения происходят относительно медленно, а оптимизационная задача решается относительно быстро. Однако для больших нестационарных оптимизационных задач решение, получаемое таким способом, оказывается далеким от оптимального в силу изменения исходных данных в процессе вычислений. В этом случае необходимо использовать алгоритмы, динамически корректирующие вычислительный процесс в соответствии с изменяющимися исходными данными. Тем самым, вычисления с измененными данными начинаются не с нуля, а используют информацию, полученную в прошлом. Такой подход применим для решения задач реального времени при условии, что алгоритм достаточно быстро отслеживает движение точки оптимума. Для больших задач ЛП последнее требование делает актуальной разработку масштабируемых методов и параллельных алгоритмов линейного программирования.

До настоящего времени одним из самых популярных способов решения задач ЛП является семейство алгоритмов, разработанных на основе симплекс-метода [16]. Симплекс-метод способен решать большие задачи ЛП, эффективно используя различные виды гиперразреженности [17]. Однако симплекс-методу присущ ряд фундаментальных недостатков. Во-первых, на некоторых задачах симплекс-методу приходится обходить все вершины симплекса, что соответствует экспоненциальной временно́й сложности [18]. Во-вторых, при решении симплекс-методом задач ЛП, размерность которых превышает 50 000, часто наблюдается потеря точности [19], которую не удается компенсировать применением даже таких мощных алгоритмов, как аффинное масштабирование или итеративное уточнение [20]. В-третьих, информационная структура алгоритмов, основанных на симплекс-методе, имеет ограниченный ресурс параллелизма, что делает невозможным их эффективное распараллеливание на больших вычислительных системах с распределенной памятью [21, 22]. Все перечисленное затрудняет использование симплекс-метода для решения нестационарных задач ЛП в режиме реального времени.

Другим популярным подходом к решению больших задач ЛП является класс алгоритмов, основанных на методе внутренних точек [23], предложенном Дикиным [24]. Эти алгоритмы способны решать задачи ЛП с миллионами переменных и ограничений [25]. Достоинством метода внутренних точек является то, что он самокорректируется и способен обеспечить высокую точность вычислений. Основными

2023, **24** (4), 408–429. doi 10.26089/NumMet.v24r428

недостатками метода внутренних точек являются следующие. Во-первых, значимый подкласс алгоритмов, основанных на методе внутренних точек, требует в качестве начального приближения некоторую внутреннюю точку допустимой области задачи ЛП. Нахождение такой точки можно свести к решению дополнительной задачи ЛП [26]. Другим способом нахождения внутренних точек является использование фейеровских приближений [27]. Во-вторых, метод внутренних точек плохо масштабируется в больших вычислительных системах кластерного типа. Известны некоторые частные случае, когда удается выполнить эффективное распараллеливание метода внутренних точек (см., например, [28]), но в общем случае построить эффективную параллельную реализацию этого метода для кластерных вычислительных систем не удается. В-третьих, итерационный характер метода внутренних точек не позволяет заранее предсказать время вычислений для конкретной задачи ЛП. Указанные недостатки затрудняют применение метода внутренних точек для решения больших задач ЛП в режиме реального времени.

Новым перспективным подходом к решению оптимизационных задач, вызывающим большой интерес, являются искусственные нейронные сети [29], представляющие собой мощный универсальный инструмент, применимый практически во всех проблемных областях. Одним из первых применений искусственных нейронных сетей к решению задач ЛП была работа Хопфилда и Танка [30]. Нейронная сеть Хопфилда–Танка состоит из двух полносвязных слоев и является рекуррентной. Число нейронов первого слоя совпадает с числом переменных задачи ЛП. Число нейронов во втором слое равно числу ограничений. Веса и смещения нейронной сети полностью определяются параметрами задачи ЛП. Выходной сигнал циклически подается на вход нейронной сети. Нейронная сеть работает до достижения состояния равновесия, когда выход становится равным входу. Это состояние соответствует минимуму специальной энергетической функции и является решением задачи ЛП. Подход Хопфилда–Танка был развит в большом количестве работ (см., например, [31–35]). Главным недостатком этого подхода является то, что невозможно предсказать количество циклов работы нейронной сети, необходимое для достижения состояния равновесия. Это делает невозможным использование таких рекуррентных сетей для решения больших задач ЛП в режиме реального времени. Для этой цели более перспективными представляются глубокие нейронные сети прямого распространения. Архитектура и параметры таких сетей, как правило, не зависят от входных данных задачи. Решение получается за один проход с фиксированным временем работы сети, что обеспечивает возможность их применения для решения задач в режиме реального времени. Особый интерес представляют сверточные нейронные сети [36], ориентированные на распознавание и классификацию образов. В недавней работе [37] был предложен оригинальный метод построения образов многомерных задач ЛП, открывающий возможность использовать нейронные сети прямого распространения, включая сверточные, для их решения. Необходимо отметить, что глубокие нейронные сети требуют обучения на большом количестве размеченных прецедентов, которое может быть эффективно выполнено на графических процессорах [38]. В статье [39] предложен итерационный апекс-метод решения задач ЛП, позволяющий построить на поверхности допустимого многогранника путь в направлении максимального увеличения/уменьшения значения целевой функции, приводящий к точке оптимума. Апекс-метод принадлежит к классу итерационных методов проективного типа, для которых характерна низкая линейная скорость сходимости, что делает их неприемлемыми для режима реального времени. Однако апекс-метод позволяет построить размеченное множество прецедентов для обучения нейронных сетей прямого распространения.

В данной статье представлен новый итерационный метод решения задач ЛП, получивший название “метод поверхностного движения”. Указанный метод ориентирован на использование искусственных нейронных сетей прямого распространения, в том числе сверточных нейронных сетей. Статья организована следующим образом. В разделе 2 представлен теоретический базис, на котором основан метод поверхностного движения. Раздел 3 посвящен описанию метода поверхностного движения и доказательству теоремы сходимости. В разделе 4 обсуждаются сильные и слабые стороны предложенного метода, а также раскрываются пути его практической реализации на основе синтеза суперкомпьютерных и нейросетевых технологий. В разделе 5 суммируются полученные результаты и приводятся направления дальнейших исследований. Сводка обозначений, используемых в статье, приведена в разделе 6.

2. Теоретический базис. В этом разделе описывается теоретический фундамент, на котором базируется метод поверхностного движения.

Сформулируем задачу ЛП в следующем виде:

$$\bar{\boldsymbol{x}} = \arg\max_{\boldsymbol{x}\in\mathbb{R}^n} \{\langle \boldsymbol{c}, \boldsymbol{x}\rangle \mid A\boldsymbol{x} \leqslant \boldsymbol{b}\}, \tag{1}$$

2023, **24** (4), 408–429. doi 10.26089/NumMet.v24r428

где $\boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{b} \in \mathbb{R}^m$, $A \in \mathbb{R}^{m\times n}$, $m > 1$, $\boldsymbol{c} \neq \boldsymbol{0}$. Здесь $\langle\cdot,\cdot\rangle$ обозначает скалярное произведение двух векторов. Мы предполагаем, что ограничение $\boldsymbol{x} \geqslant \boldsymbol{0}$ также включено в матричное неравенство $A\boldsymbol{x} \leqslant \boldsymbol{b}$ в форме

$$-\boldsymbol{x} \leqslant \boldsymbol{0}.$$

Обозначим через $\mathcal{P}$ множество индексов, нумерующих строки матрицы A:

$$\mathcal{P} = \{1, \ldots, m\}.$$

Линейная целевая функция задачи (1) имеет вид

$$f(\boldsymbol{x}) = \langle \boldsymbol{c}, \boldsymbol{x}\rangle.$$

Вектор $\boldsymbol{c}$ в данном случае является градиентом целевой функции $f(\boldsymbol{x})$.

Пусть $\boldsymbol{a}_i \in \mathbb{R}^n$ обозначает вектор, представляющий i-ю строку матрицы A. Мы предполагаем, что $\boldsymbol{a}_i \neq \boldsymbol{0}$ для всех $i \in \mathcal{P}$. Обозначим через $\hat{H}_i$ замкнутое полупространство, определяемое неравенством $\langle \boldsymbol{a}_i, \boldsymbol{x}\rangle \leqslant b_i$, а через H_i — ограничивающую его гиперплоскость:

$$\hat{H}_i = \{\boldsymbol{x} \in \mathbb{R}^n \,|\, \langle \boldsymbol{a}_i, \boldsymbol{x}\rangle \leqslant b_i\}, \tag{2}$$

$$H_i = \{\boldsymbol{x} \in \mathbb{R}^n \,|\, \langle \boldsymbol{a}_i, \boldsymbol{x}\rangle = b_i\}. \tag{3}$$

Определим допустимый многогранник

$$M = \bigcap_{i\in\mathcal{P}} \hat{H}_i, \tag{4}$$

представляющий множество допустимых точек задачи ЛП (1). Заметим, что M в этом случае будет замкнутым выпуклым множеством. Мы будем предполагать, что множество M является ограниченным и $M \neq \emptyset$, т.е. задача ЛП (1) имеет решение. Обозначим через $\Gamma(M)$ множество граничных точек многогранника M[1].

Утверждение 1. *Пусть M — допустимый многогранник задачи ЛП (1), определяемый формулой (4). Тогда для любой точки $\boldsymbol{u} \in \Gamma(M)$ существует $\varepsilon > 0$ такое, что для любой граничной точки $\boldsymbol{w}$, принадлежащей ε-окрестности $V_\varepsilon(\boldsymbol{u})$ точки $\boldsymbol{u}$, найдется $i' \in \mathcal{P}$, для которого справедливо $\boldsymbol{u}, \boldsymbol{w} \in H_{i'}$:*

$$\forall \boldsymbol{u} \in \Gamma(M)\ \exists \varepsilon > 0 : \forall \boldsymbol{w} \in V_\varepsilon(\boldsymbol{u}) \cap \Gamma(M)\ \exists i' \in \mathcal{P} : \boldsymbol{u}, \boldsymbol{w} \in H_{i'}.$$

Доказательство. Зафиксируем произвольную точку $\boldsymbol{u} \in \Gamma(M)$. Обозначим

$$\mathcal{P}_{\boldsymbol{u}} = \{\, i \in \mathcal{P} |\, \boldsymbol{u} \in H_i\}. \tag{5}$$

Другими словами, $\mathcal{P}_{\boldsymbol{u}}$ — множество индексов всех гиперплоскостей H_i, которым принадлежит точка $\boldsymbol{u}$. Положим

$$\mathcal{P}_{\setminus \boldsymbol{u}} = \mathcal{P} \setminus \mathcal{P}_{\boldsymbol{u}},$$

т.е. $\mathcal{P}_{\setminus \boldsymbol{u}}$ — множество индексов всех гиперплоскостей H_i, которым точка $\boldsymbol{u}$ не принадлежит. Определим

$$\delta = \min\left\{ \operatorname{dist}(\boldsymbol{u}, H_i) |\, i \in \mathcal{P}_{\setminus \boldsymbol{u}}\right\},$$

где $\operatorname{dist}(\boldsymbol{u}, H_i)$ обозначает евклидово расстояние от точки $\boldsymbol{u}$ до гиперплоскости H_i[2]. По определению

$$\delta > 0.$$

Возьмем ε, удовлетворяющее условию

$$0 < \varepsilon < \delta.$$

[1] Под граничной точкой множества $M \subset \mathbb{R}^n$ понимается точка в $\mathbb{R}^n$, для которой любая открытая ее окрестность в $\mathbb{R}^n$ имеет непустое пересечение как с множеством M, так и с его дополнением.

[2] В данном случае $\operatorname{dist}(\boldsymbol{u}, H_i) = \dfrac{\langle \boldsymbol{a}_i, \boldsymbol{u}\rangle - b_i}{\|\boldsymbol{a}_i\|}$.

2023, **24** (4), 408–429. doi 10.26089/NumMet.v24r428

Тогда для любого $\boldsymbol{w} \in V_\varepsilon(\boldsymbol{u}) \cap \Gamma(M)$ имеем

$$\forall i \in \mathcal{P}_{\setminus \boldsymbol{u}} : \boldsymbol{w} \notin H_i.$$

Поскольку $\boldsymbol{w}$ — граничная точка, то отсюда следует, что найдется $i' \in \mathcal{P}_{\boldsymbol{u}}$ такой, что

$$\boldsymbol{w} \in H_{i'}.$$

Заметим, что в силу (5) также имеем

$$\boldsymbol{u} \in H_{i'}.$$

Утверждение 1 доказано.

Определение 1. *Целевой проекцией точки $\boldsymbol{z} \in \mathbb{R}^n$ на гиперплоскость H_i называется точка $\boldsymbol{\gamma}_i(\boldsymbol{z}) \in \mathbb{R}^n \cup \{\infty\}$, определяемая формулой*

$$\boldsymbol{\gamma}_i(\boldsymbol{z}) = \begin{cases} L(\boldsymbol{z}) \cap H_i, & \text{если } \langle \boldsymbol{a}_i, \boldsymbol{c} \rangle \neq 0; \\ \infty, & \text{если } \langle \boldsymbol{a}_i, \boldsymbol{c} \rangle = 0, \end{cases} \tag{6}$$

где $L(\boldsymbol{z})$ — прямая, проходящая через точку $\boldsymbol{z}$ параллельно вектору $\boldsymbol{c}$:

$$L(\boldsymbol{z}) = \{\, \boldsymbol{y} \in \mathbb{R}^n \,|\, \boldsymbol{y} = \boldsymbol{z} + \lambda \boldsymbol{c}, \lambda \in \mathbb{R} \,\}. \tag{7}$$

Другими словами, если вектор $\boldsymbol{c}$ не параллелен гиперплоскости H_i, то целевой проекцией точки $\boldsymbol{z}$ на гиперплоскость H_i является точка пересечения этой гиперплоскости с прямой, проходящей через точку $\boldsymbol{z}$ параллельно вектору $\boldsymbol{c}$ (см. рис. 1). В случае, когда вектор $\boldsymbol{c}$ параллелен гиперплоскости H_i, целевая проекция полагается равной бесконечно удаленной точке ∞.

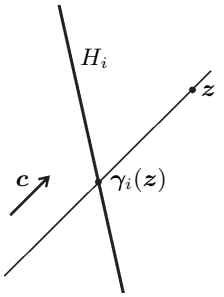

Рис. 1. Целевая проекция $\boldsymbol{\gamma}_i(\boldsymbol{z})$ точки $\boldsymbol{z}$ на гиперплоскость H_i

Fig. 1. Objective projection $\boldsymbol{\gamma}_i(\boldsymbol{z})$ of point $\boldsymbol{z}$ onto hyperplane H_i

Следующее утверждение предоставляет формулу для вычисления целевой проекции $\boldsymbol{\gamma}_i(\boldsymbol{z})$ точки $\boldsymbol{z}$ на гиперплоскость H_i.

Утверждение 2. *Пусть $\langle \boldsymbol{a}_i, \boldsymbol{c} \rangle \neq 0$. Тогда*

$$\boldsymbol{\gamma}_i(\boldsymbol{z}) = \boldsymbol{z} - \frac{\langle \boldsymbol{a}_i, \boldsymbol{z} \rangle - b_i}{\langle \boldsymbol{a}_i, \boldsymbol{c} \rangle} \boldsymbol{c}. \tag{8}$$

Доказательство. В соответствии с (6) и (7) имеем

$$\boldsymbol{\gamma}_i(\boldsymbol{z}) = \boldsymbol{z} + \lambda \boldsymbol{c} \tag{9}$$

при некотором $\lambda \in \mathbb{R}$. С другой стороны, в соответствии с (3) имеем

$$\langle \boldsymbol{a}_i, \boldsymbol{\gamma}_i(\boldsymbol{z}) \rangle = b_i. \tag{10}$$

Подставим правую часть формулы (9) в формулу (10) вместо $\boldsymbol{\gamma}_i(z)$:

$$\langle \boldsymbol{a}_i, \boldsymbol{z} + \lambda \boldsymbol{c} \rangle = b_i.$$

Отсюда

$$\lambda = -\frac{\langle \boldsymbol{a}_i, \boldsymbol{z} \rangle - b_i}{\langle \boldsymbol{a}_i, \boldsymbol{c} \rangle}. \tag{11}$$

Подставив правую часть формулы (11) вместо λ в формулу (9), получаем

$$\boldsymbol{\gamma}_i(\boldsymbol{z}) = \boldsymbol{z} - \frac{\langle \boldsymbol{a}_i, \boldsymbol{z} \rangle - b_i}{\langle \boldsymbol{a}_i, \boldsymbol{c} \rangle} \boldsymbol{c}.$$

Утверждение 2 доказано.

2023, **24** (4), 408–429. doi 10.26089/NumMet.v24r428

Определение 2. *Целевым смещением точки* $\boldsymbol{z} \in \mathbb{R}^n$ *относительно гиперплоскости* H_i *называется скалярная величина* $\beta_i(\boldsymbol{z})$*, вычисляемая по формуле*

$$\beta_i(\boldsymbol{z}) = -\frac{\langle \boldsymbol{a}_i, \boldsymbol{z}\rangle - b_i}{\langle \boldsymbol{a}_i, \boldsymbol{c}\rangle} \|\boldsymbol{c}\|. \tag{12}$$

Далее мы для краткости будем использовать термин “смещение”, подразумевая под этим “целевое смещение”. Обозначим

$$\boldsymbol{e_c} = \frac{\boldsymbol{c}}{\|\boldsymbol{c}\|}. \tag{13}$$

Тогда формулу (8) можно переписать в виде

$$\boldsymbol{\gamma}_i(\boldsymbol{z}) = \boldsymbol{z} + \beta_i(\boldsymbol{z})\boldsymbol{e_c}, \tag{14}$$

что равносильно

$$\beta_i(\boldsymbol{z})\boldsymbol{e_c} = \boldsymbol{\gamma}_i(\boldsymbol{z}) - \boldsymbol{z}.$$

С учетом (13) отсюда следует

$$|\beta_i(\boldsymbol{z})| = \|\boldsymbol{\gamma}_i(\boldsymbol{z}) - \boldsymbol{z}\|.$$

Таким образом, $|\beta_i(\boldsymbol{z})|$ является расстоянием от точки $\boldsymbol{z}$ до ее целевой проекции на гиперплоскость H_i.

Определение 3. *Целевой гиперплоскостью* $H_c(\boldsymbol{z})$*, проходящей через точку* $\boldsymbol{z}$*, будем называть гиперплоскость, задаваемую формулой*

$$H_c(\boldsymbol{z}) = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{c}, \boldsymbol{x}\rangle = \langle \boldsymbol{c}, \boldsymbol{z}\rangle\}. \tag{15}$$

Справедливо следующее утверждение.

Утверждение 3. *Зафиксируем произвольную точку* $\boldsymbol{z} \in \mathbb{R}^n$*. Тогда для любых точек* $\boldsymbol{z}'$, $\boldsymbol{z}'' \in H_c(\boldsymbol{z})$, $\boldsymbol{z}' \neq \boldsymbol{z}''$ *справедливо*

$$\langle \boldsymbol{c}, \boldsymbol{\gamma}_i(\boldsymbol{z}')\rangle < \langle \boldsymbol{c}, \boldsymbol{\gamma}_i(\boldsymbol{z}'')\rangle \Leftrightarrow \beta_i(\boldsymbol{z}') < \beta_i(\boldsymbol{z}'')$$

для всех $i \in \mathcal{P}$.

Доказательство. Поскольку $\boldsymbol{z}', \boldsymbol{z}'' \in H_c(\boldsymbol{z})$, а $\boldsymbol{c}$ является нормалью к гиперплоскости $H_c(\boldsymbol{z})$, справедливы следующие два равенства:

$$\langle \boldsymbol{c}, \boldsymbol{z}' - \boldsymbol{z}\rangle = 0,$$

$$\langle \boldsymbol{c}, \boldsymbol{z}'' - \boldsymbol{z}\rangle = 0.$$

Следовательно,

$$\langle \boldsymbol{c}, \boldsymbol{z}''\rangle = \langle \boldsymbol{c}, \boldsymbol{z}\rangle = \langle \boldsymbol{c}, \boldsymbol{z}'\rangle. \tag{16}$$

Последовательно используя формулы (14), (16) и (13), получаем следующую цепочку эквивалентных неравенств:

$$\begin{aligned}
\langle \boldsymbol{c}, \boldsymbol{\gamma}_i(\boldsymbol{z}')\rangle < \langle \boldsymbol{c}, \boldsymbol{\gamma}_i(\boldsymbol{z}'')\rangle &\Leftrightarrow \langle \boldsymbol{c}, \boldsymbol{z}' + \beta_i(\boldsymbol{z}')\boldsymbol{e_c}\rangle < \langle \boldsymbol{c}, \boldsymbol{z}'' + \beta_i(\boldsymbol{z}'')\boldsymbol{e_c}\rangle \\
&\Leftrightarrow \langle \boldsymbol{c}, \boldsymbol{z}'\rangle + \langle \boldsymbol{c}, \beta_i(\boldsymbol{z}')\boldsymbol{e_c}\rangle < \langle \boldsymbol{c}, \boldsymbol{z}''\rangle + \langle \boldsymbol{c}, \beta_i(\boldsymbol{z}'')\boldsymbol{e_c}\rangle \\
&\Leftrightarrow \langle \boldsymbol{c}, \beta_i(\boldsymbol{z}')\boldsymbol{e_c}\rangle < \langle \boldsymbol{c}, \beta_i(\boldsymbol{z}'')\boldsymbol{e_c}\rangle \\
&\Leftrightarrow \langle \boldsymbol{c}, \beta_i(\boldsymbol{z}')\boldsymbol{c}/\|\boldsymbol{c}\|\rangle < \langle \boldsymbol{c}, \beta_i(\boldsymbol{z}'')\boldsymbol{c}/\|\boldsymbol{c}\|\rangle \\
&\Leftrightarrow \frac{\beta_i(\boldsymbol{z}')}{\|\boldsymbol{c}\|}\langle \boldsymbol{c}, \boldsymbol{c}\rangle < \frac{\beta_i(\boldsymbol{z}'')}{\|\boldsymbol{c}\|}\langle \boldsymbol{c}, \boldsymbol{c}\rangle \\
&\Leftrightarrow \beta_i(\boldsymbol{z}') < \beta_i(\boldsymbol{z}'').
\end{aligned}$$

Утверждение 3 доказано.

Следуя [39], дадим определение рецессивного полупространства.

Определение 4. *Полупространство* $\hat{H}_i$ *называется рецессивным, если*

$$\forall \boldsymbol{x} \in H_i, \forall \lambda > 0 : \boldsymbol{x} + \lambda \boldsymbol{c} \notin \hat{H}_i. \tag{17}$$

2023, **24** (4), 408–429. doi 10.26089/NumMet.v24r428

Геометрический смысл этого определения состоит в том, что луч, исходящий в направлении вектора $\boldsymbol{c}$ из любой точки гиперплоскости, ограничивающей рецессивное полупространство, не имеет общих точек с этим полупространством, за исключением начальной. Известно [39], что следующее условие является необходимым и достаточным для того, чтобы полупространство $\hat{H}_i$ было рецессивным:

$$\langle \boldsymbol{a}_i, \boldsymbol{c} \rangle > 0. \tag{18}$$

Рецессивное полупространство обладает следующими свойствами.

Свойство 1. *Пусть полупространство $\hat{H}_i$ является рецессивным. Тогда любая прямая, параллельная вектору $\boldsymbol{c}$, пересекает гиперплоскость H_i в единственной точке.*

Данное свойство непосредственно вытекает из того факта, что гиперплоскость H_i, ограничивающая рецессивное полупространство $\hat{H}_i$, по определению не может быть параллельна вектору $\boldsymbol{c}$.

Свойство 2. *Пусть полупространство $\hat{H}_i$ является рецессивным. Тогда*

$$\boldsymbol{x} \in \hat{H}_i \Leftrightarrow \beta_i(\boldsymbol{x}) \geqslant 0.$$

Доказательство. Сначала предположим, что $\boldsymbol{x} \in \hat{H}_i$. Тогда в соответствии с (2) справедливо неравенство

$$\langle \boldsymbol{a}_i, \boldsymbol{x} \rangle - b_i \leqslant 0.$$

В силу (12) имеем

$$\beta_i(\boldsymbol{x}) = -\frac{\langle \boldsymbol{a}_i, \boldsymbol{x} \rangle - b_i}{\langle \boldsymbol{a}_i, \boldsymbol{c} \rangle} \|\boldsymbol{c}\|. \tag{19}$$

Принимая во внимание (18), получаем

$$\boldsymbol{x} \in \hat{H}_i \Rightarrow \beta_i(\boldsymbol{x}) \geqslant 0.$$

Теперь предположим, что $\beta_i(\boldsymbol{x}) \geqslant 0$. В соответствии с (19) это означает, что

$$\frac{\langle \boldsymbol{a}_i, \boldsymbol{x} \rangle - b_i}{\langle \boldsymbol{a}_i, \boldsymbol{c} \rangle} \|\boldsymbol{c}\| \leqslant 0.$$

Учитывая (18), отсюда получаем

$$\langle \boldsymbol{a}_i, \boldsymbol{x} \rangle - b_i \leqslant 0.$$

Согласно (2), отсюда следует, что

$$\boldsymbol{x} \in \hat{H}_i.$$

Таким образом,

$$\beta_i(\boldsymbol{x}) \geqslant 0 \;\Rightarrow\; \boldsymbol{x} \in \hat{H}_i.$$

Свойство 2 доказано.

Определим

$$\mathcal{I} = \{ i \in \mathcal{P} \mid \langle \boldsymbol{a}_i, \boldsymbol{c} \rangle > 0 \}, \tag{20}$$

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

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

Положим

$$\hat{M} = \bigcap_{i \in \mathcal{I}} \hat{H}_i. \tag{21}$$

Очевидно, что $\hat{M}$ является выпуклым замкнутым неограниченным многогранником. Будем называть его рецессивным. Из (4) и (20) следует

$$M \subset \hat{M}.$$

Обозначим через $\Gamma(\hat{M})$ множество граничных точек рецессивного многогранника $\hat{M}$. Согласно утверждению 3 в [39] имеем

$$\bar{\boldsymbol{x}} \in \Gamma(\hat{M}),$$

т.е. решение задачи ЛП (1) лежит на границе рецессивного многогранника $\hat{M}$.

2023, **24** (4), 408–429. doi 10.26089/NumMet.v24r428

Утверждение 4. *Пусть $\hat{M}$ — рецессивный многогранник, определяемый формулой (21). Тогда для любой точки $\boldsymbol{u} \in \Gamma(\hat{M})$ существует $\varepsilon > 0$ такое, что для любой граничной точки $\boldsymbol{w}$, принадлежащей ε-окрестности $V_\varepsilon(\boldsymbol{u})$ точки $\boldsymbol{u}$, найдется $i' \in \mathcal{I}$, для которого справедливо $\boldsymbol{u}, \boldsymbol{w} \in H_{i'}$:*

$$\forall \boldsymbol{u} \in \Gamma(\hat{M})\ \exists \varepsilon > 0 : \forall \boldsymbol{w} \in V_\varepsilon(\boldsymbol{u}) \cap \Gamma(\hat{M})\ \exists i' \in \mathcal{I} : \boldsymbol{u}, \boldsymbol{w} \in H_{i'}.$$

Доказательство. Доказательство идентично доказательству утверждения 1.

Определение 5. *Целевой проекцией точки $\boldsymbol{z} \in \mathbb{R}^n$ на границу $\Gamma(\hat{M})$ рецессивного многогранника $\hat{M}$ называется точка $\hat{\boldsymbol{\gamma}}(\boldsymbol{z})$, вычисляемая по формуле*

$$\hat{\boldsymbol{\gamma}}(\boldsymbol{z}) = L(\boldsymbol{z}) \cap \Gamma(\hat{M}),$$

где $L(\boldsymbol{z})$ — прямая, проходящую через точку $\boldsymbol{z}$ параллельно вектору $\boldsymbol{c}$:

$$L(\boldsymbol{z}) = \{\, \boldsymbol{y} \in \mathbb{R}^n \,|\, \boldsymbol{y} = \boldsymbol{z} + \lambda \boldsymbol{c}, \lambda \in \mathbb{R} \}.$$

Скалярную величину $\hat{\beta}(\boldsymbol{z}) \in \mathbb{R}$, удовлетворяющую уравнению

$$\hat{\boldsymbol{\gamma}}(\boldsymbol{z}) = \boldsymbol{z} + \hat{\beta}(\boldsymbol{z})\boldsymbol{c}, \tag{22}$$

будем называть смещением точки $\boldsymbol{z}$ относительно границы рецессивного многогранника $\hat{M}$.

Заметим, что корректность этого определения базируется на свойстве 1. Следующее утверждение предоставляет формулу для вычисления целевой проекции на границу рецессивного многогранника.

Утверждение 5. *Пусть задана произвольная точка $\boldsymbol{z} \in \mathbb{R}^n$. Положим*

$$i' = \arg\min \{\, \beta_i(\boldsymbol{z}) \,|\, i \in \mathcal{I} \,\}. \tag{23}$$

Тогда

$$\hat{\boldsymbol{\gamma}}(\boldsymbol{z}) = \boldsymbol{\gamma}_{i'}(\boldsymbol{z}). \tag{24}$$

Другими словами, целевая проекция точки $\boldsymbol{z}$ на границу рецессивного многогранника $\hat{M}$ совпадает с проекцией этой точки на гиперплоскость $H_{i'}$, имеющей минимальное смещение относительно $\boldsymbol{z}$.

Доказательство. Зафиксируем произвольную точку $\boldsymbol{z} \in \mathbb{R}^n$. В соответствии с определением 5 построим прямую, параллельную вектору $\boldsymbol{c}$, которая проходит через точку $\boldsymbol{z}$:

$$L = \{\, \boldsymbol{y} \in \mathbb{R}^n \,|\, \boldsymbol{y} = \boldsymbol{z} + \lambda \boldsymbol{c}, \lambda \in \mathbb{R} \}.$$

Имеем

$$\hat{\boldsymbol{\gamma}}(\boldsymbol{z}) = L \cap \Gamma(\hat{M}).$$

Согласно свойству 1 для любого $i \in \mathcal{I}$ прямая L пересекает H_i только в одной точке, которую мы обозначим через $\boldsymbol{y}_i$:

$$L \cap H_i = \{\boldsymbol{y}_i\}.$$

Определим

$$Y = \bigcup_{i \in \mathcal{I}} \{\boldsymbol{y}_i\},$$

т.е. Y — множество точек, в которых прямая L пересекает границы рецессивных полупространств. По определению 1 имеем

$$\forall i \in \mathcal{I}\ :\ \boldsymbol{\gamma}_i(\boldsymbol{z}) = \boldsymbol{y}_i \tag{25}$$

и

$$\forall i \in \mathcal{I}\ :\ \boldsymbol{y}_i \in H_i. \tag{26}$$

В силу (21) также имеем

$$\hat{\boldsymbol{\gamma}}(\boldsymbol{z}) \in Y.$$

Это означает, что найдется $i' \in \mathcal{I}$ такой, что

$$\hat{\boldsymbol{\gamma}}(\boldsymbol{z}) = \boldsymbol{\gamma}_{i'}(\boldsymbol{z}). \tag{27}$$

2023, **24** (4), 408–429. doi 10.26089/NumMet.v24r428

Покажем, что

$$i' \in \operatorname{Arg\,min}\{\beta_i(\boldsymbol{z}) \,|\, i \in \mathcal{I}\}.$$

Предположим противное, т.е.

$$\beta_{i'}(\boldsymbol{z}) > \min\{\beta_i(\boldsymbol{z}) \,|\, i \in \mathcal{I}\}.$$

Тогда существует $i'' \in \mathcal{I}$ такой, что

$$\beta_{i''}(\boldsymbol{z}) < \beta_{i'}(\boldsymbol{z}). \tag{28}$$

В силу (14) и (25) имеем

$$\boldsymbol{y}_{i'} = \boldsymbol{z} + \beta_{i'}(\boldsymbol{z})\boldsymbol{e}_{\boldsymbol{c}},$$
$$\boldsymbol{y}_{i''} = \boldsymbol{z} + \beta_{i''}(\boldsymbol{z})\boldsymbol{e}_{\boldsymbol{c}}.$$

Отсюда

$$\boldsymbol{y}_{i'} = \boldsymbol{y}_{i''} + (\beta_{i'}(\boldsymbol{z}) - \beta_{i''}(\boldsymbol{z}))\,\boldsymbol{e}_{\boldsymbol{c}},$$

что в силу (28) и (13) равносильно

$$\boldsymbol{y}_{i'} = \boldsymbol{y}_{i''} + \frac{|\beta_{i''}(\boldsymbol{z}) - \beta_{i'}(\boldsymbol{z})|}{\|\boldsymbol{c}\|}\boldsymbol{c}. \tag{29}$$

В соответствии с (17) и (26), из (29) следует

$$\boldsymbol{y}_{i'} \notin \hat{H}_{i''}.$$

Это означает, что

$$\boldsymbol{y}_{i'} \notin \hat{M}.$$

Принимая во внимание (25) и (27), отсюда следует

$$\hat{\boldsymbol{\gamma}}(\boldsymbol{z}) \notin \hat{M}.$$

Получили противоречие с определением 5. Утверждение 5 доказано.

Следующее утверждение предоставляет формулу для вычисления смещения точки относительно границы рецессивного многогранника.

Утверждение 6. *Пусть задана произвольная точка $\boldsymbol{z} \in \mathbb{R}^n$. Тогда*

$$\hat{\beta}(\boldsymbol{z}) = \frac{\langle \boldsymbol{c}, \hat{\boldsymbol{\gamma}}(\boldsymbol{z}) - \boldsymbol{z}\rangle}{\|\boldsymbol{c}\|^2}. \tag{30}$$

Доказательство. В соответствии с (22) точки $\hat{\boldsymbol{\gamma}}(\boldsymbol{z})$ и $\boldsymbol{z}$ находятся на нормали к гиперплоскости $H_c(\boldsymbol{z})$. Поэтому точка $\boldsymbol{z} \in H_c(\boldsymbol{z})$ является ортогональной проекцией точки $\hat{\boldsymbol{\gamma}}(\boldsymbol{z})$ на гиперплоскость $H_c(\boldsymbol{z})$ и с учетом (15) может быть вычислена по известной формуле

$$\boldsymbol{z} = \hat{\boldsymbol{\gamma}}(\boldsymbol{z}) - \frac{\langle \boldsymbol{c}, \hat{\boldsymbol{\gamma}}(\boldsymbol{z}) - \boldsymbol{z}\rangle}{\|\boldsymbol{c}\|^2}\boldsymbol{c}.$$

Перепишем это в виде

$$\hat{\boldsymbol{\gamma}}(\boldsymbol{z}) = \boldsymbol{z} + \frac{\langle \boldsymbol{c}, \hat{\boldsymbol{\gamma}}(\boldsymbol{z}) - \boldsymbol{z}\rangle}{\|\boldsymbol{c}\|^2}\boldsymbol{c}.$$

Сопоставляя это с (22), получаем

$$\hat{\beta}(\boldsymbol{z}) = \frac{\langle \boldsymbol{c}, \hat{\boldsymbol{\gamma}}(\boldsymbol{z}) - \boldsymbol{z}\rangle}{\|\boldsymbol{c}\|^2}.$$

Утверждение 6 доказано.

3. Метод поверхностного движения. Метод поверхностного движения строит на поверхности допустимого многогранника путь из произвольной граничной точки $\boldsymbol{u}^{(0)} \in M \cap \Gamma(\hat{M})$ до точки $\bar{\boldsymbol{x}}$, являющейся решением задачи ЛП (1). Перемещение по поверхности рецессивного многогранника происходит в направлении наибольшего увеличения значения целевой функции. Путь, построенный в результате такого движения, будем называть оптимальным целевым путем. Реализация метода поверхностного движения приведена в виде алгоритма 1. Прокомментируем шаги этого алгоритма.

На шаге 1 вводится начальное приближение $\boldsymbol{u}^{(0)}$. Им может быть произвольная граничная точка рецессивного многогранника $\hat{M}$, удовлетворяющая условию

$$\boldsymbol{u}^{(0)} \in M \cap \Gamma(\hat{M}).$$

2023, **24** (4), 408–429. doi 10.26089/NumMet.v24r428

Алгоритм 1. Метод поверхностного движения
Algorithm 1. Surface movement method

Require $\hat{H}_i = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_i, \boldsymbol{x} \rangle \leqslant b_i\}$, $\hat{M} = \bigcap\limits_{i \in \mathcal{I}} \hat{H}_i$, $H_c(\boldsymbol{z}) = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{c}, \boldsymbol{x} - \boldsymbol{z} \rangle = 0\}$

1: **input** $\boldsymbol{u}^{(0)}$
2: **assert** $\boldsymbol{u}^{(0)} \in M \cap \Gamma(\hat{M})$
3: $k := 0$
4: $r := 0.1$
5: $D^{(0)} := H_c(\boldsymbol{u}^{(0)}) \cap V_r(\boldsymbol{u}^{(0)})$ // $V_r(\boldsymbol{u}^{(0)})$ — r-neighborhood of the point $\boldsymbol{u}^{(0)}$
6: $\boldsymbol{v}^{(0)} := \arg\max\{\hat{\beta}(\boldsymbol{z}) \mid \boldsymbol{z} \in D^{(0)}\}$
7: $\boldsymbol{w}^{(0)} := \hat{\boldsymbol{\gamma}}(\boldsymbol{v}^{(0)})$
8: **assert** $\exists i \in \mathcal{I} : \boldsymbol{w}^{(0)}, \boldsymbol{u}^{(0)} \in H_i \cap \Gamma(\hat{M})$ // if not satisfied, reduce r
9: **while** $\langle \boldsymbol{c}, \boldsymbol{w}^{(k)} - \boldsymbol{u}^{(k)} \rangle > \varepsilon_f$ **do**
10: **assert** $\exists i \in \mathcal{I} : \boldsymbol{w}^{(k)}, \boldsymbol{u}^{(k)} \in H_i \cap \Gamma(\hat{M})$ // if not satisfied, reduce r
11: $\boldsymbol{d}^{(k)} := \boldsymbol{w}^{(k)} - \boldsymbol{u}^{(k)}$
12: $L^{(k)} := \{\boldsymbol{u}^{(k)} + \lambda \boldsymbol{d}^{(k)} \mid \lambda \in \mathbb{R}_{>0}\}$
13: $\boldsymbol{u}^{(k+1)} := \arg\max\{\|\boldsymbol{x} - \boldsymbol{u}^{(k)}\| \mid \boldsymbol{x} \in L^{(k)} \cap \Gamma(M)\}$
14: $D^{(k+1)} := H_c(\boldsymbol{u}^{(k+1)}) \cap V_r(\boldsymbol{u}^{(k+1)})$
15: $\boldsymbol{v}^{(k+1)} := \arg\max\{\hat{\beta}(\boldsymbol{z}) \mid \boldsymbol{z} \in D^{(k+1)}\}$
16: $\boldsymbol{w}^{(k+1)} := \hat{\boldsymbol{\gamma}}(\boldsymbol{v}^{(k+1)})$
17: $k := k + 1$
18: **end while**
19: **output** $\boldsymbol{u}^{(k)}$
20: **stop**

Это условие проверяется на шаге 2. Для получения подходящего начального приближения может применяться алгоритм, реализующий стадию Quest апекс-метода [39]. На шаге 3 счетчик итераций k устанавливается в 0. На шаге 4 задается начальное значение параметра r. На шаге 5 строится n-мерный диск $D^{(0)}$, являющийся пересечением целевой гиперплоскости $H_c\left(\boldsymbol{u}^{(0)}\right)$, проходящей через точку $\boldsymbol{u}^{(0)}$, и n-мерного шара $V_r\left(\boldsymbol{u}^{(0)}\right)$ малого радиуса r с центром в точке $\boldsymbol{u}^{(0)}$. На шаге 6 вычисляется точка $\boldsymbol{v}^{(0)} \in D^{(0)}$ с максимальным смещением относительно границы рецессивного многогранника $\hat{M}$.

Смещение $\hat{\beta}(\boldsymbol{z})$ вычисляется с помощью формулы (30). Целевая проекция $\hat{\boldsymbol{\gamma}}(\boldsymbol{z})$, используемая в формуле (30), вычисляется с помощью формул (23), (24) и (8). Смещение $\beta_i(\boldsymbol{z})$, используемое в формуле (23), вычисляется с помощью формулы (12). Шаг 7 вычисляет точку $\boldsymbol{w}^{(0)}$, являющуюся целевой проекцией точки $\boldsymbol{v}^{(0)}$ на границу рецессивного многогранника. Шаг 8 проверяет, что существует рецессивное полупространство $\hat{H}_i$ такое, что граничные точки $\boldsymbol{w}^{(0)}$ и $\boldsymbol{u}^{(0)}$ лежат на грани $H_i \cap \Gamma(\hat{M})$. Это необходимо для того, чтобы перемещение происходило по поверхности рецессивного многогранника, а не через его внутреннюю область. Если указанное требование не выполняется, необходимо уменьшить радиус r n-мерного шара $V_r\left(\boldsymbol{u}^{(0)}\right)$. Подходящий r найдется в силу утверждения 4. Шаги 9–18 реализуют основной цикл метода поверхностного движения, геометрическая интерпретация которого приведена на рис. 2. Этот цикл выполняется пока справедливо условие

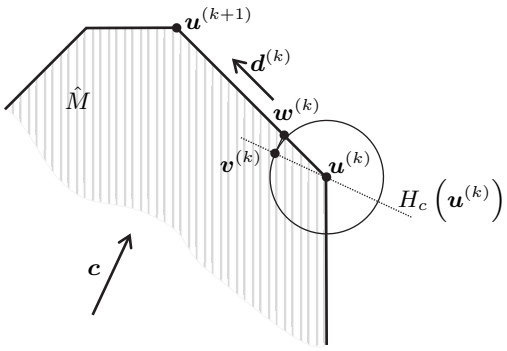

Рис. 2. Метод поверхностного движения
Fig. 2. Surface movement method

$$\left\langle \boldsymbol{c}, \boldsymbol{w}^{(k)} - \boldsymbol{u}^{(k)} \right\rangle > \varepsilon_f, \tag{31}$$

где ε_f — малый положительный параметр. Шаг 10 проверяет, что существует рецессивное полупространство $\hat{H}_i$ такое, что граничные точки $\boldsymbol{w}^{(k)}$ и $\boldsymbol{u}^{(k)}$ лежат на грани $H_i \cap \Gamma(\hat{M})$. Если указанное требование не выполняется, необходимо уменьшить радиус r n-мерного шара $V_r\left(\boldsymbol{u}^{(k)}\right)$. На шаге 11 формируется век-

2023, **24** (4), 408–429. doi 10.26089/NumMet.v24r428

тор $\boldsymbol{d}^{(k)}$, определяющий направление движения. Шаг 12 строит луч $L^{(k)}$ с началом в точке $\boldsymbol{u}^{(k)}$, сонаправленный с вектором $\boldsymbol{d}^{(k)}$. На шаге 13 определяется следующее приближение $\boldsymbol{u}^{(k+1)}$ как точка на луче $L^{(k)}$, лежащая на границе допустимого многогранника M и максимально удаленная от точки $\boldsymbol{u}^{(k)}$. Шаг 14 строит гипердиск $D^{(k+1)}$ радиуса r с центром в точке $\boldsymbol{u}^{(k+1)}$, лежащий на гиперплоскости $H_c\left(\boldsymbol{u}^{(k+1)}\right)$. Шаг 15 находит на гипердиске $D^{(k+1)}$ точку $\boldsymbol{v}^{(k+1)}$ с максимальным смещением. На шаге 16 вычисляется точка $\boldsymbol{w}^{(k+1)}$, являющаяся целевой проекцией точки $\boldsymbol{v}^{(k+1)}$ на границу рецессивного многогранника $\hat{M}$. Шаг 17 увеличивает счетчик итераций k на единицу. На шаге 18 происходит переход на начало цикла **while**. Шаг 19 выводит в качестве результата последнее приближение $\boldsymbol{u}^{(k)}$. Шаг 20 завершает работу алгоритма.

Отметим, что по построению алгоритма 1 для любого k имеет место

$$\boldsymbol{u}^{(k)} \in M \cap \Gamma(\hat{M}), \tag{32}$$

т.е. все точки последовательности $\left\{\boldsymbol{u}^{(k)}\right\}$, генерируемой алгоритмом 1, одновременно лежат и на границе допустимого многогранника M, и на границе рецессивного многогранника $\hat{M}$. Кроме этого, из (31) следует

$$\left\langle \boldsymbol{c}, \boldsymbol{u}^{(k)} \right\rangle < \left\langle \boldsymbol{c}, \boldsymbol{w}^{(k)} \right\rangle. \tag{33}$$

Также справедливо неравенство

$$\left\langle \boldsymbol{c}, \boldsymbol{w}^{(k)} \right\rangle \leqslant \left\langle \boldsymbol{c}, \boldsymbol{u}^{(k+1)} \right\rangle. \tag{34}$$

Следующая лемма гарантирует завершение работы алгоритма 1 за конечное число итераций.

Лемма 1. *Пусть допустимый многогранник M задачи ЛП (1) является ограниченным непустым множеством. Тогда последовательность точек $\left\{\boldsymbol{u}^{(k)}\right\}$, генерируемая алгоритмом 1, является конечной для любого $\varepsilon_f > 0$.*

Доказательство. Предположим противное, т.е. алгоритм 1 генерирует бесконечную последовательность точек $\left\{\boldsymbol{u}^{(k)}\right\}_{k=0}^{\infty}$. Но тогда, в силу (33) и (34), мы получаем бесконечную строго возрастающую числовую последовательность $\left\{\left\langle \boldsymbol{c}, \boldsymbol{u}^{(k)} \right\rangle\right\}_{k=0}^{\infty}$:

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

Так как по условию леммы допустимый многогранник M является непустым ограниченным множеством, задача ЛП (1) имеет решение $\bar{\boldsymbol{x}}$. В силу (32) имеем

$$\left\langle \boldsymbol{c}, \boldsymbol{u}^{(k)} \right\rangle \leqslant \langle \boldsymbol{c}, \bar{\boldsymbol{x}} \rangle$$

для всех $k = 0, 1, 2, \ldots$. Это означает, что последовательность $\left\{\left\langle \boldsymbol{c}, \boldsymbol{u}^{(k)} \right\rangle\right\}_{k=0}^{\infty}$ является ограниченной сверху. По теореме Вейерштрасса монотонно возрастающая ограниченная сверху числовая последовательность имеет конечный предел, равный ее супремуму. Поэтому существует $k' \in \mathbb{N}$ такой, что

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

В силу (34) отсюда следует

$$\forall k > k' : \left\langle \boldsymbol{c}, \boldsymbol{w}^{(k)} \right\rangle - \left\langle \boldsymbol{c}, \boldsymbol{u}^{(k)} \right\rangle < \varepsilon_f,$$

что равносильно

$$\forall k > k' : \left\langle \boldsymbol{c}, \boldsymbol{w}^{(k)} - \boldsymbol{u}^{(k)} \right\rangle < \varepsilon_f.$$

Получили противоречие с условием (31) выполнения цикла, используемым на шаге 9 алгоритма 1. Лемма 1 доказана.

Следующая теорема показывает, что при достаточно малом ε_f результатом работы алгоритма 1 будет решение задачи ЛП (1).

Теорема 1. *Пусть допустимый многогранник M задачи ЛП (1) является ограниченным непустым множеством. Пусть $\bar{\boldsymbol{x}}$ — решение задачи ЛП (1). Пусть задана монотонно убывающая последовательность положительных чисел $\{\varepsilon_\eta\}_{\eta=1}^{\infty}$, стремящаяся к нулю:*

$$\lim_{\eta \to \infty} \varepsilon_\eta = 0. \tag{35}$$

2023, **24** (4), 408–429. doi 10.26089/NumMet.v24r428

Обозначим через $u^{(K_\eta)}$ *конечную точку, генерируемую алгоритмом 1 при* $\varepsilon_f = \varepsilon_\eta$ *(она существует в силу леммы 1). Тогда найдется* $\bar{\eta} \in \mathbb{N}$ *такое, что для всех* $\eta \geqslant \bar{\eta}$

$$\left\langle \boldsymbol{c}, \boldsymbol{u}^{(K_\eta)} \right\rangle = \langle \boldsymbol{c}, \bar{\boldsymbol{x}} \rangle .$$

Доказательство. Покажем, что последовательность $\left\{ \left\langle \boldsymbol{c}, \boldsymbol{u}^{(K_\eta)} \right\rangle \right\}_{\eta=1}^{\infty}$ является монотонно возрастающей. Действительно, из (33) и (34) следует, что

$$\forall k' \leqslant k'' : \left\langle \boldsymbol{c}, \boldsymbol{u}^{(k')} \right\rangle \leqslant \left\langle \boldsymbol{c}, \boldsymbol{u}^{(k'')} \right\rangle . \tag{36}$$

По условию теоремы

$$\varepsilon_\eta \geqslant \varepsilon_{\eta+1}.$$

Следовательно, по построению алгоритма 1

$$K_\eta \leqslant K_{\eta+1}.$$

Сопоставляя это с (36), получаем

$$\left\langle \boldsymbol{c}, \boldsymbol{u}^{(K_\eta)} \right\rangle \leqslant \left\langle \boldsymbol{c}, \boldsymbol{u}^{(K_{\eta+1})} \right\rangle ,$$

т.е. последовательность $\left\{ \left\langle \boldsymbol{c}, \boldsymbol{u}^{(K_\eta)} \right\rangle \right\}_{\eta=1}^{\infty}$ является монотонно возрастающей. Очевидно, что эта последовательность ограничена сверху величиной $\langle \boldsymbol{c}, \bar{\boldsymbol{x}} \rangle$. Следовательно, она имеет конечный предел:

$$\lim_{\eta \to \infty} \left\langle \boldsymbol{c}, \boldsymbol{u}^{(K_\eta)} \right\rangle = \bar{f}.$$

Алгоритм 1 в рамках каждой итерации проходит[3] одну грань/ребро рецессивного многогранника $\hat{M}$ в направлении максимального увеличения значения целевой функции. При этом каждая грань/ребро проходится не более одного раза, так как многогранник $\hat{M}$ является выпуклым множеством. Это означает, что существует $\bar{\eta} \in \mathbb{N}$ такое, что для всех $\eta \geqslant \bar{\eta}$ имеем

$$\boldsymbol{u}^{(K_\eta)} = \boldsymbol{u}^{(K_{\bar{\eta}})}$$

и

$$\left\langle \boldsymbol{c}, \boldsymbol{u}^{(K_\eta)} \right\rangle = \bar{f}.$$

По построению алгоритма 1, учитывая (35), это возможно только в том случае, когда

$$\left\langle \boldsymbol{c}, \boldsymbol{w}^{(K_{\bar{\eta}})} - \boldsymbol{u}^{(K_{\bar{\eta}})} \right\rangle = 0. \tag{37}$$

Покажем, что в этом случае

$$\left\langle \boldsymbol{c}, \boldsymbol{u}^{(K_{\bar{\eta}})} \right\rangle = \langle \boldsymbol{c}, \bar{\boldsymbol{x}} \rangle ,$$

т.е. точка $\boldsymbol{u}^{(K_{\bar{\eta}})}$ является решением задачи ЛП (1). Обозначим $\boldsymbol{u}' = \boldsymbol{u}^{(K_{\bar{\eta}})}$. Предположим противное: существует точка

$$\boldsymbol{u}'' \in M \tag{38}$$

такая, что

$$\langle \boldsymbol{c}, \boldsymbol{u}'' \rangle > \langle c, \boldsymbol{u}' \rangle .$$

Это равносильно

$$\langle \boldsymbol{c}, \boldsymbol{u}'' - \boldsymbol{u}' \rangle > 0. \tag{39}$$

[3]Под прохождением грани/ребра многогранника понимается движение внутри линейного многообразия размерности k при наличии k степеней свободы ($0 < k < n$).

2023, **24** (4), 408–429. doi 10.26089/NumMet.v24r428

Рис. 3 иллюстрирует последующую часть доказательства. Исходя из определения 3, вычислим ортогональную проекцию $\boldsymbol{p}$ точки $\boldsymbol{u}''$ на целевую гиперплоскость $H_c(\boldsymbol{u}')$, проходящую через точку $\boldsymbol{u}'$:

$$\boldsymbol{p} = \boldsymbol{u}'' - \frac{\langle \boldsymbol{c}, \boldsymbol{u}'' - \boldsymbol{u}' \rangle}{\|\boldsymbol{c}\|^2} \boldsymbol{c}. \tag{40}$$

Заметим, что

$$\|\boldsymbol{p} - \boldsymbol{u}'\| \neq 0, \tag{41}$$

так как в противном случае, в соответствии с определением 4, точка $\boldsymbol{u}''$ не может принадлежать рецессивному многограннику $\hat{M}$, что противоречит формуле (38). Выберем $r \in \mathbb{R}_0$, удовлетворяющий условию

$$r > 0, \tag{42}$$

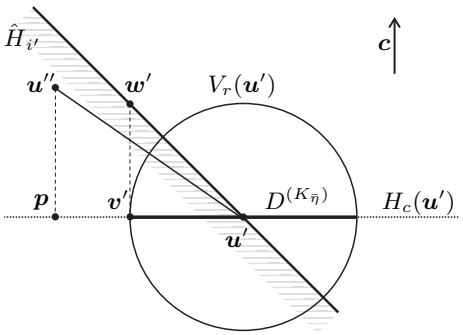

Рис. 3. Иллюстрация к доказательству теоремы 1

Fig. 3. Illustration to proof of Theorem 1

для которого существует $i' \in \mathcal{I}$ такой, что

$$\hat{\boldsymbol{\gamma}}(\boldsymbol{v}'), \boldsymbol{u}' \in H_{i'} \cap \Gamma(M), \tag{43}$$

где

$$\boldsymbol{v}' = \boldsymbol{u}' + \frac{r}{\|\boldsymbol{p} - \boldsymbol{u}'\|}(\boldsymbol{p} - \boldsymbol{u}'). \tag{44}$$

Это возможно в силу утверждения 1. Без ограничения общности мы можем считать, что r удовлетворяет всем проверкам **assert** в случае, когда $\varepsilon_f = \varepsilon_{\bar{\eta}}$. Таким образом,

$$\boldsymbol{v}' \in D^{(K_{\bar{\eta}})},$$

где $D^{K_{\bar{\eta}}}$ — диск, вычисляемый на шаге 14 алгоритма 1 при $k = K_{\bar{\eta}} - 1$. Отметим, что

$$i' = \arg\min \{\beta_i(\boldsymbol{v}') \,|\, i \in \mathcal{I}\},$$

так как в противном случае $\hat{\boldsymbol{\gamma}}(\boldsymbol{v}') \notin H_{i'}$, что противоречит (43). Согласно утверждению 5 отсюда следует

$$\hat{\boldsymbol{\gamma}}(\boldsymbol{v}') = \boldsymbol{\gamma}_{i'}(\boldsymbol{v}'). \tag{45}$$

Положим

$$\boldsymbol{w}' = \hat{\boldsymbol{\gamma}}(\boldsymbol{v}').$$

С учетом (45) имеем

$$\boldsymbol{w}' = \boldsymbol{\gamma}_{i'}(\boldsymbol{v}').$$

Используя утверждение 2, отсюда получаем

$$\boldsymbol{w}' = \boldsymbol{v}' - \frac{\langle \boldsymbol{a}_{i'}, \boldsymbol{v}' \rangle - b_{i'}}{\langle \boldsymbol{a}_{i'}, \boldsymbol{c} \rangle} \boldsymbol{c}. \tag{46}$$

Поскольку $\boldsymbol{u}' \in H_{i'}$, из (3) следует

$$\langle \boldsymbol{a}_{i'}, \boldsymbol{u}' \rangle = b_{i'}. \tag{47}$$

Поэтому (46) можно переписать в виде

$$\boldsymbol{w}' = \boldsymbol{v}' - \frac{\langle \boldsymbol{a}_{i'}, \boldsymbol{v}' \rangle - \langle \boldsymbol{a}_{i'}, \boldsymbol{u}' \rangle}{\langle \boldsymbol{a}_{i'}, \boldsymbol{c} \rangle} \boldsymbol{c}.$$

Подставив вместо $\boldsymbol{v}'$ правую часть формулы (44), отсюда получаем

$$\boldsymbol{w}' = \boldsymbol{u}' + \frac{r}{\|\boldsymbol{p} - \boldsymbol{u}'\|}(\boldsymbol{p} - \boldsymbol{u}') - \frac{\left\langle \boldsymbol{a}_{i'}, \boldsymbol{u}' + \dfrac{r}{\|\boldsymbol{p} - \boldsymbol{u}'\|}(\boldsymbol{p} - \boldsymbol{u}') \right\rangle - \langle \boldsymbol{a}_{i'}, \boldsymbol{u}' \rangle}{\langle \boldsymbol{a}_{i'}, \boldsymbol{c} \rangle} \boldsymbol{c},$$

2023, **24** (4), 408–429. doi 10.26089/NumMet.v24r428

что равносильно

$$\boldsymbol{w}' = \boldsymbol{u}' + \frac{r}{\|\boldsymbol{p}-\boldsymbol{u}'\|}(\boldsymbol{p}-\boldsymbol{u}') - \frac{\left\langle \boldsymbol{a}_{i'}, \frac{r}{\|\boldsymbol{p}-\boldsymbol{u}'\|}(\boldsymbol{p}-\boldsymbol{u}') \right\rangle}{\langle \boldsymbol{a}_{i'}, \boldsymbol{c}\rangle}\boldsymbol{c}. \tag{48}$$

В соответствии с (2) имеем

$$\hat{H}_{i'} = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_{i'}, \boldsymbol{x}\rangle \leqslant b_{i'}\}. \tag{49}$$

Используя (47), формулу (49) можно переписать в виде

$$\hat{H}_{i'} = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_{i'}, \boldsymbol{x}\rangle \leqslant \langle \boldsymbol{a}_{i'}, \boldsymbol{u}'\rangle\}. \tag{50}$$

Из (38) следует $\boldsymbol{u}'' \in \hat{H}_{i'}$. Сопоставляя это с (50), получаем

$$\langle \boldsymbol{a}_{i'}, \boldsymbol{u}''\rangle \leqslant \langle \boldsymbol{a}_{i'}, \boldsymbol{u}'\rangle,$$

что равносильно

$$\langle \boldsymbol{a}_{i'}, \boldsymbol{u}' - \boldsymbol{u}''\rangle \geqslant 0. \tag{51}$$

Поскольку полупространство $\hat{H}_{i'}$ является рецессивным, в соответствии с утверждением 1 в [39] справедливо

$$\langle \boldsymbol{a}_{i'}, \boldsymbol{c}\rangle > 0. \tag{52}$$

В силу (48) и (40) имеем

$$\langle \boldsymbol{c}, \boldsymbol{w}' - \boldsymbol{u}'\rangle = \left\langle \boldsymbol{c}, \frac{r}{\|\boldsymbol{p}-\boldsymbol{u}'\|}(\boldsymbol{p}-\boldsymbol{u}') - \frac{\left\langle \boldsymbol{a}_{i'}, \frac{r}{\|\boldsymbol{p}-\boldsymbol{u}'\|}(\boldsymbol{p}-\boldsymbol{u}') \right\rangle}{\langle \boldsymbol{a}_{i'}, \boldsymbol{c}\rangle}\boldsymbol{c} \right\rangle =$$

$$= \frac{r}{\|\boldsymbol{p}-\boldsymbol{u}'\|}\left(\langle \boldsymbol{c}, \boldsymbol{p}-\boldsymbol{u}'\rangle - \frac{\langle \boldsymbol{a}_{i'}, \boldsymbol{p}-\boldsymbol{u}'\rangle}{\langle \boldsymbol{a}_{i'}, \boldsymbol{c}\rangle}\|\boldsymbol{c}\|\right) =$$

$$= \frac{r}{\|\boldsymbol{p}-\boldsymbol{u}'\|}\left(\left\langle \boldsymbol{c}, \boldsymbol{u}'' - \frac{\langle \boldsymbol{c}, \boldsymbol{u}''-\boldsymbol{u}'\rangle}{\|\boldsymbol{c}\|^2}\boldsymbol{c} - \boldsymbol{u}' \right\rangle - \frac{\left\langle \boldsymbol{a}_{i'}, \boldsymbol{u}'' - \frac{\langle \boldsymbol{c}, \boldsymbol{u}''-\boldsymbol{u}'\rangle}{\|\boldsymbol{c}\|^2}\boldsymbol{c} - \boldsymbol{u}' \right\rangle}{\langle \boldsymbol{a}_{i'}, \boldsymbol{c}\rangle}\|\boldsymbol{c}\|\right) =$$

$$= \frac{r}{\|\boldsymbol{p}-\boldsymbol{u}'\|}\left(-\frac{\left\langle \boldsymbol{a}_{i'}, \boldsymbol{u}'' - \frac{\langle \boldsymbol{c}, \boldsymbol{u}''-\boldsymbol{u}'\rangle}{\|\boldsymbol{c}\|^2}\boldsymbol{c} - \boldsymbol{u}' \right\rangle}{\langle \boldsymbol{a}_{i'}, \boldsymbol{c}\rangle}\|\boldsymbol{c}\|\right) =$$

$$= \frac{r\|\boldsymbol{c}\|}{\|\boldsymbol{p}-\boldsymbol{u}'\|\langle \boldsymbol{a}_{i'}, \boldsymbol{c}\rangle}\left(-\left\langle \boldsymbol{a}_{i'}, \boldsymbol{u}'' - \frac{\langle \boldsymbol{c}, \boldsymbol{u}''-\boldsymbol{u}'\rangle}{\|\boldsymbol{c}\|^2}\boldsymbol{c} - \boldsymbol{u}' \right\rangle\right) =$$

$$= \frac{r\|\boldsymbol{c}\|}{\|\boldsymbol{p}-\boldsymbol{u}'\|\langle \boldsymbol{a}_{i'}, \boldsymbol{c}\rangle}\left(\langle \boldsymbol{a}_{i'}, \boldsymbol{u}'-\boldsymbol{u}''\rangle + \frac{\langle \boldsymbol{c}, \boldsymbol{u}''-\boldsymbol{u}'\rangle\langle \boldsymbol{a}_{i'}, \boldsymbol{c}\rangle}{\|\boldsymbol{c}\|^2}\right).$$

В соответствии с (42), (41), (52), (51) и (39) отсюда следует

$$\langle \boldsymbol{c}, \boldsymbol{w}' - \boldsymbol{u}'\rangle > 0.$$

Вспомнив, что $\boldsymbol{u}' = \boldsymbol{u}^{(K_{\bar{\eta}})}$, перепишем последнее неравенство в виде

$$\left\langle \boldsymbol{c}, \boldsymbol{w}' - \boldsymbol{u}^{(K_{\bar{\eta}})} \right\rangle > 0. \tag{53}$$

2023, **24** (4), 408–429. doi 10.26089/NumMet.v24r428

Алгоритм 2. Метод ЛП с использованием DNN

Algorithm 2. LP method using DNN

Require $\hat{H}_i = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_i, \boldsymbol{x} \rangle \leqslant b_i\}$, $\hat{M} = \bigcap\limits_{i \in \mathcal{I}} \hat{H}_i$, $H_c(\boldsymbol{z}) = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{c}, \boldsymbol{x} - \boldsymbol{z} \rangle = 0\}$

1: **input** $\boldsymbol{u}^{(0)}$; **assert** $\boldsymbol{u}^{(0)} \in M \cap \Gamma(\hat{M})$
2: $k := 0$
3: $D^{(0)} := \mathfrak{G}(\boldsymbol{u}^{(0)})$
4: $\boldsymbol{d}^{(0)} := \mathrm{DNN}(D^{(0)})$
5: **while** $\boldsymbol{d}^{(k)} \neq \boldsymbol{0}$ **do**
6: $L^{(k)} := \{\boldsymbol{u}^{(k)} + \lambda \boldsymbol{d}^{(k)} \mid \lambda \in \mathbb{R}_{>0}\}$
7: $\boldsymbol{u}^{(k+1)} := \arg\max\{\|\boldsymbol{x} - \boldsymbol{u}^{(k)}\| \mid \boldsymbol{x} \in L^{(k)} \cap \Gamma(M)\}$
8: $D^{(k+1)} := \mathfrak{G}(\boldsymbol{u}^{(k+1)})$
9: $\boldsymbol{d}^{(k+1)} := \mathrm{DNN}(D^{(k+1)})$
10: $k := k + 1$
11: **end while**
12: **output** $\boldsymbol{u}^{(k)}$
13: **stop**

В силу утверждения 3 по построению (см. шаги 15, 16 алгоритма 1) имеем

$$\left\langle \boldsymbol{c}, \boldsymbol{w}^{(K_{\bar{\eta}})} \right\rangle \geqslant \langle \boldsymbol{c}, \boldsymbol{w}' \rangle .$$

Сопоставляя это с (53), получаем

$$\left\langle \boldsymbol{c}, \boldsymbol{w}^{(K_{\bar{\eta}})} - \boldsymbol{u}^{(K_{\bar{\eta}})} \right\rangle > 0,$$

что противоречит (37). Теорема 1 доказана.

4. Обсуждение. В данном разделе обсуждаются сильные и слабые стороны предложенного метода, а также раскрываются пути его практической реализации на основе синтеза суперкомпьютерных и нейросетевых технологий.

Прежде всего рассмотрим возможность реализации алгоритма 1 в виде программы для ЭВМ. На шаге 15 алгоритма 1 необходимо найти точку гипердиска $D^{(k+1)}$, имеющую максимальное смещение. Нам неизвестен алгоритм, позволяющий получить численное решение этой задачи. Однако мы видим следующий путь решения, предполагающий использование искусственной нейронной сети. Применяя подход, описанный в статье [37], мы заменяем гипердиск $D^{(k+1)}$ на регулярное множество точек, называемое рецептивным полем. Каждой точке рецептивного поля мы сопоставляем ее смещение относительно границы допустимого многогранника. В результате получаем матрицу размерности $(n-1)$, представляющую собой локальный образ задачи ЛП. Локальность образа означает, что мы получаем визуальное представление поверхности не всего допустимого многогранника, а только некоторой его части в окрестности точки текущего приближения. Этот образ подается на вход предварительно обученной нейронной сети прямого распространения, которая определяет вектор $\boldsymbol{d}$, указывающий направление движения на поверхности допустимого многогранника в сторону максимального увеличения значения целевой функции. Обозначим через $\mathfrak{G}(\boldsymbol{u})$ функцию, строящую рецептивное поле с центром в точке $\boldsymbol{u}$ и вычисляющую локальный образ задачи ЛП в этой точке. Детально алгоритм построения многомерного образа задачи ЛП описан и исследован в работе [37]. Обозначим через DNN глубокую нейронную сеть прямого распространения, на вход которой подается локальный образ задачи ЛП, а на выходе получается вектор $\boldsymbol{d}$, задающий направление поверхностного движения. Тогда алгоритм 1 может быть преобразован в алгоритм 2, допускающий реализацию на практике. Множество размеченных прецедентов, необходимое для обучения DNN, может быть получено с помощью апекс-метода [39], строящего путь, близкий к оптимальному целевому пути[4].

[4]Оптимальный целевой путь — это путь на поверхности допустимого многогранника из начальной точки в направлении максимального увеличения значения целевой функции.

2023, **24** (4), 408–429. doi 10.26089/NumMet.v24r428

Дадим оценку временно́й сложности алгоритма 2. Метод поверхностного движения посещает[5] каждую гиперплоскость рецессивного многогранника не более одного раза. Посещение гиперплоскости выполняется в пределах одной итерации цикла **while** (шаги 5–11 алгоритма 2). Следовательно, общее количество итераций можно оценить как $O(m)$, где m — количество ограничений задачи ЛП (1). Нахождение следующего приближения $\boldsymbol{u}^{(k+1)}$ можно реализовать путем дихотомии. Таким образом, количество операций для шага 7 не зависит ни от количества ограничений m, ни от размерности n, и для больших значений m и n может быть приближенно оценено как константа. Наиболее трудоемким является построение локального образа задачи ЛП на шаге 8 алгоритма 2. Рецептивное поле в виде гиперкубической решетки состоит из $\eta^{(n-1)}$ точек, где n — размерность пространства, а η — количество точек по одному измерению. Однако последние исследования [40] показали, что крестообразное рецептивное поле с количеством точек $(\eta-1)n+1$ дает результаты, не уступающие гиперкубическому по точности решения задачи ЛП. Предварительно обученная нейронная сеть прямого распространения DNN вычисляет вектор движения $\boldsymbol{d}^{(k+1)}$ на шаге 9 за время, зависящее только от n, так как на вход ей подается $(\eta-1)n+1$ чисел (в случае крестообразного рецептивного поля). Таким образом, временна́я сложность алгоритма в целом может быть оценена как $O(mn)$. В дополнение отметим, что граница масштабируемости[6] алгоритма построения визуального образа задачи ЛП может быть оценена как $O\left(\sqrt{2n^2m+m^2n+8nm-6m}\right)$ [37]. Если предположить, что $m=O(n)$, то для границы масштабируемости мы получаем оценку $O\left(n\sqrt{n}\right)$, близкую к линейной зависимости. Это означает, что алгоритм построения локального образа задачи ЛП может быть эффективно распараллелен на большом количестве процессорных узлов кластерной вычислительной системы. Так, для $n=7$ и $m=15$ вычислительные эксперименты показали пик ускорения на 326 процессорных узлах [37]. Заметим, что количество итераций алгоритма 2 не зависит от ε_f, так как такой параметр в этом алгоритме отсутствует.

Метод поверхностного движения является итерационным и самокорректирующимся. Поэтому он может потенциально применяться для решения нестационарных задач. При этом, если меняется только целевая функция, то алгоритм 2 вообще не требует никаких принципиальных изменений. Важно, чтобы скорость корректировки опережала скорость изменений. Если же меняется система ограничений (без изменения размерности), то алгоритм 2 потребует определенных модификаций, так как текущее приближение может “погрузиться” в многогранник или “оторваться” от его поверхности. Авторы планируют детально исследовать этот вопрос в будущем.

Алгоритм 2 также может применяться для решения задач ЛП в режиме реального времени. Действительно, количество итераций ограничено параметром m. При фиксированном n построение локального образа задачи ЛП требует фиксированного количества операций. При этом процедура построения образа эффективно распараллеливается на большом количестве процессорных узлов. Искусственная нейронная сеть прямого распространения DNN анализирует локальный образ задачи ЛП за фиксированное время, зависящее только от n. Работа нейронной сети также может быть эффективно распараллелена с помощью графических процессоров. В перспективе можно отказаться от путешествия по граням/ребрам допустимого многогранника и анализировать с помощью нейронной сети образ всего многогранника, полученный из точки апекса (см. [39]). Нейронная сеть будет выдавать приближенное решение, которое может быть уточнено путем анализа фиксированного числа локальных образов с увеличивающимся масштабом. Однако этот вопрос нуждается в дальнейших исследованиях.

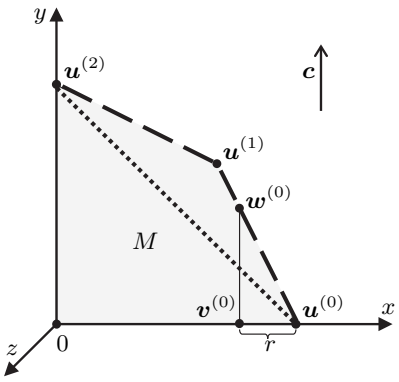

Рис. 4. Оптимальный целевой путь обозначен пунктиром; путь минимальной длины показан точками

Fig. 4. Optimal objective path is indicated by dashed line; path of minimum length is shown by dotted line

[5]Под посещением гиперплоскости понимается прямолинейное движение от точки входа на гиперплоскость до точки первого изменения направления движения.

[6]Под границей масштабируемости понимается число процессорных узлов кластерной вычислительной системы, на котором достигается максимум ускорения.

2023, **24** (4), 408–429. doi 10.26089/NumMet.v24r428

Алгоритм 1 строит на поверхности допустимого многогранника оптимальный целевой путь к решению задачи ЛП. Это непосредственно следует из построения алгоритма и утверждения 3. Интересен вопрос, всегда ли этот путь является путем наименьшей длины в смысле евклидовой метрики. Следующий простой пример в пространстве $\mathbb{R}^3$, проиллюстрированный на рис. 4, показывает, что это не всегда так. Для системы ограничений

$$\begin{cases} x + 2y \leqslant 2, \\ 2x + y \leqslant 2, \\ x \geqslant 0, \\ y \geqslant 0, \\ z = 0 \end{cases}$$

и целевой функции $f(x, y, z) = y$ оптимальный целевой путь, обозначенный пунктиром, может не совпадать с кратчайшим путем, обозначенным точками.

Также может возникнуть вопрос, можно ли на шаге 13 алгоритма 1 заменить $\Gamma(M)$ на $\Gamma(\hat{M})$, так как это сокращает количество неравенств, вовлекаемых в проверку условия $x \in \Gamma(M)$. Ответ — отрицательный. Например, в пространстве $\mathbb{R}^2$ для системы ограничений

$$\begin{cases} x + 2y \leqslant 2, \\ 2x + y \leqslant 2, \\ x + y \geqslant 1, \\ x \geqslant 0, \\ y \geqslant 0 \end{cases}$$

и целевой функции $f(x, y) = y$ для $\boldsymbol{u}^{(1)} = \left(\frac{2}{3}, \frac{2}{3}\right)$ получаем

$$\max\left\{ \|\boldsymbol{x} - \boldsymbol{u}^{(1)}\| \mid \boldsymbol{x} \in L^{(1)} \cap \Gamma(\hat{M}) \right\} = +\infty$$

(см. рис. 5).

В качестве недостатка метода поверхностного движения можно отметить то, что он аффинно не инвариантен, так как функционал цены отождествляется с вектором. Таким образом, поведение метода зависит от евклидовой структуры, определенной системой координат.

Научная новизна и теоретическая значимость предложенного метода заключается в том, что он впервые открывает возможность применения искусственных нейронных сетей прямого распространения для решения многомерных задач ЛП на основе анализа их образов.

В настоящее время идет работа над реализацией представленного метода в виде программного комплекса для кластерных вычислительных систем. Этот программный комплекс включает в себя параллельный алгоритм генерации многомерных локальных образов задач ЛП, параллельный алгоритм генерации размеченных прецедентов для обучения нейронной сети и нейросетевые модели для различных размерностей. По завершении реализации мы планируем провести сравнение метода поверхностного движения с другими методами решения задач ЛП, в частности с симплекс-методом. Описанию указанного программного комплекса и анализу результатов вычислительных экспериментов будет посвящена отдельная научная статья.

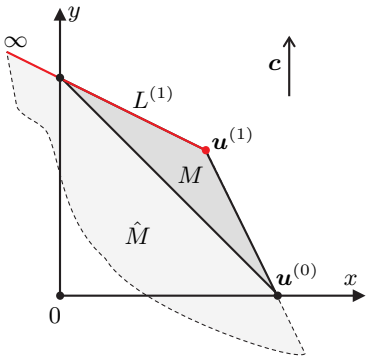

Рис. 5. Оптимальный целевой путь уходит из $\boldsymbol{u}^{(1)}$ в бесконечность на рецессивном многограннике $\hat{M}$

Fig. 5. Optimal objective path goes from $\boldsymbol{u}^{(1)}$ to infinity on recessive polytope $\hat{M}$

5. Заключение. В статье описан новый метод решения задачи ЛП, получивший название “метод поверхностного движения”. Указанный метод строит на поверхности многогранника, ограничивающего допустимую область задачи ЛП, путь от начальной точки до точки решения задачи ЛП. Вектор движения

2023, **24** (4), 408–429. doi 10.26089/NumMet.v24r428

всегда выбирается в направлении максимального увеличения/уменьшения значения целевой функции. Получившийся путь называется оптимальным целевым путем.

Метод поверхностного движения предполагает использование глубокой нейронной сети прямого распространения для определения направления движения по граням допустимого многогранника. Для этого строится многомерный локальный образ задачи ЛП в точке текущего приближения, который подается на вход нейронной сети. Множество размеченных прецедентов, необходимое для обучения нейронной сети, может быть получено с помощью апекс-метода.

Для построения теоретического фундамента метода поверхностного движения введено понятие целевой проекции — косоугольной проекции в направлении, параллельном вектору градиента целевой функции. Определена скалярная величина, называемая смещением. Модуль смещения равен расстоянию от точки до ее целевой проекции. Знак смещения определяется положением точки внутри или вне допустимого многогранника. Получена формула вычисления смещения точки относительно границы допустимого многогранника. Показано, что большему смещению соответствует большее значение целевой функции. Приведено формализованное описание метода поверхностного движения в виде алгоритма. Доказана основная теорема сходимости метода поверхностного движения к решению задачи ЛП за конечное число итераций. Приведен вариант алгоритма поверхностного движения, использующий функцию построения локального многомерного образа задачи ЛП и глубокую нейронную сеть.

В качестве направлений дальнейших исследований можно указать следующие.

1. Разработка и обучение сети DNN, способной вычислять вектор движения в направлении максимального увеличения значения целевой функции для многомерных задач ЛП.
2. Разработка программного комплекса для кластерной вычислительной системы, реализующего алгоритм 2 путем синтеза суперкомпьютерных и нейросетевых технологий.
3. Исследование зависимости точности работы сети DNN от плотности рецептивного поля.
4. Исследование применимости метода поверхностного движения для решения нестационарных задач ЛП.
5. Исследование применимости метода поверхностного движения для решения задач ЛП в режиме реального времени.
6. Разработка и исследование нового визуального метода решения задач ЛП с помощью нейронных сетей на основе анализа образа допустимого многогранника в целом.

6. Обозначения.

$\mathbb{R}^n$ — вещественное евклидово пространство.
$\|\cdot\|$ — евклидова норма.
$\langle\cdot,\cdot\rangle$ — скалярное произведение двух векторов.
$f(\boldsymbol{x})$ — линейная целевая функция.
$\boldsymbol{c}$ — градиент целевой функции $f(\boldsymbol{x})$.
$\boldsymbol{e_c}$ — единичный вектор, сонаправленный с вектором $\boldsymbol{c}$.
$\bar{\boldsymbol{x}}$ — решение задачи ЛП.
$\boldsymbol{a}_i$ — i-я строка матрицы A.
H_i — гиперплоскость, определяемая формулой $\langle \boldsymbol{a}_i, \boldsymbol{x}\rangle = b_i$.
$\hat{H}_i$ — полупространство, определяемое формулой $\langle \boldsymbol{a}_i, \boldsymbol{x}\rangle \leqslant b_i$.
$\mathcal{P}$ — множество индексов строк матрицы A.
M — допустимый многогранник, определяемый формулой $M = \bigcap_{i\in\mathcal{P}} \hat{H}_i$.
$\Gamma(M)$ — множество граничных точек допустимого многогранника M.
$\mathcal{I}$ — множество индексов, для которых полупространство $\hat{H}_i$ является рецессивным.
$\hat{M}$ — рецессивный многогранник, определяемый формулой $\hat{M} = \bigcap_{i\in\mathcal{I}} \hat{H}_i$.
$\Gamma(\hat{M})$ — множество граничных точек рецессивного многогранника $\hat{M}$.
$\boldsymbol{\gamma}_i(\boldsymbol{z})$ — целевая проекция точки $\boldsymbol{z}$ на гиперплоскость H_i.
$\beta_i(\boldsymbol{z})$ — целевое смещение точки $\boldsymbol{z}$ относительно гиперплоскости H_i: $|\beta_i(\boldsymbol{z})| = \|\boldsymbol{\gamma}_i(\boldsymbol{z}) - \boldsymbol{z}\|$.
$\hat{\boldsymbol{\gamma}}(\boldsymbol{z})$ — целевая проекция точки $\boldsymbol{z}$ на границу $\Gamma(\hat{M})$ рецессивного многогранника $\hat{M}$.
$\hat{\beta}(\boldsymbol{z})$ — целевое смещение точки $\boldsymbol{z}$ относительно границы $\Gamma(\hat{M})$: $|\hat{\beta}(\boldsymbol{z})| = \|\hat{\boldsymbol{\gamma}}(\boldsymbol{z}) - \boldsymbol{z}\|$.
$V_r(\boldsymbol{x})$ — гипершар радиуса r с центром в точке $\boldsymbol{x}$.

2023, **24** (4), 408–429. doi 10.26089/NumMet.v24r428

Список литературы

1. *Hartung T.* Making big sense from big data // Frontiers in Big Data. 2018. **1**. Article Number 5. doi 10.3389/fdata.2018.00005.
2. *Соколинская И.М., Соколинский Л.Б.* О решении задачи линейного программирования в эпоху больших данных // Параллельные вычислительные технологии (ПаВТ'2017). Короткие статьи и описания плакатов. Челябинск: Издательский центр ЮУрГУ, 2017. 471–484. http://omega.sp.susu.ru/pavt2017/short/014.pdf. Дата обращения: 11 ноября 2023.
3. *Branke J.* Optimization in dynamic environments // Evolutionary Optimization in Dynamic Environments. Genetic Algorithms and Evolutionary Computation. Vol. 3. Boston: Springer, 2002. 13–29. doi 10.1007/978-1-4615-0911-0_2.
4. *Ерёмин И.И., Мазуров В.Д.* Нестационарные процессы математического программирования. М.: Наука, 1979.
5. *Brogaard J., Hendershott T., Riordan R.* High-frequency trading and price discovery // Review of Financial Studies. 2014. **27**, N 8. 2267–2306. doi 10.1093/rfs/hhu032.
6. *Deng S., Huang X., Wang J., et al.* A decision support system for trading in apple futures market using predictions fusion // IEEE Access. 2021. **9**. 1271–1285. doi 10.1109/ACCESS.2020.3047138.
7. *Seregin G.* Lecture notes on regularity theory for the Navier–Stokes equations. Singapore: World Scientific Publishing Company, 2014. doi 10.1142/9314.
8. *Demin D.* Synthesis of optimal control of technological processes based on a multialternative parametric description of the final state // Eastern-European Journal of Enterprise Technologies. 2017. **3**, N 4. 51–63.
9. *Kazarinov L.S., Shnayder D.A., Kolesnikova O.V.* Heat load control in steam boilers // Proc. 2017 International Conference on Industrial Engineering, Applications and Manufacturing. St. Petersburg, 2017. doi 10.1109/ICIEAM.2017.8076177.
10. *Zagoskina E.V., Barbasova T.A., Shnaider D.A.* Intelligent control system of blast-furnace melting efficiency // Proc. Int. Multi-Conference on Engineering, Computer and Information Sciences. Novosibirsk, 2019. doi 10.1109/SIBIRCON48586.2019.8958221.
11. *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. **15**, N 4. 573–583. doi 10.1049/ITR2.12047.
12. *Scholl M., Minnerup K., Reiter C., et al.* Optimization of a thermal management system for battery electric vehicles // Proc. 14th Int. Conf. on Ecological Vehicles and Renewable Energies. Monte-Carlo, Monaco, 2019. doi 10.1109/EVER.2019.8813657.
13. *Meisel S.* Dynamic vehicle routing // Anticipatory Optimization for Dynamic Decision Making. Operations Research/Computer Science Interfaces Series. Vol. 51. New York: Springer, 2011. 77–96. doi 10.1007/978-1-4614-0505-4_6.
14. *Kiran D.R.* Production planning and control: a comprehensive approach. Oxford: Butterworth-Heinemann, 2019. doi 10.1016/C2018-0-03856-6.
15. *Mall R.* Real-time systems: theory and practice. Delhi: Pearson Education, 2007.
16. *Dantzig G.B.* Linear programming and extensions. Princeton: Princeton Univ. Press, 1998.
17. *Hall J.A.J., McKinnon K.I.M.* Hyper-sparsity in the revised simplex method and how to exploit it // Computational Optimization and Applications. 2005. **32**, N 3. 259–283. doi 10.1007/s10589-005-4802-0.
18. *Klee V., Minty G.* How good is the simplex algorithm? // Inequalities III. New York: Academic Press, 1972. 159–175.
19. *Bartels R.H., Stoer J., Zenger Ch.* A realization of the simplex method based on triangular decompositions // Handbook for Automatic Computation. Volume II: Linear Algebra. Berlin: Springer, 1971. 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. Hoboken: Wiley, 2014. 157–188. doi 10.1002/9781119005216.ch7.
21. *Hall J.A.J.* Towards a practical parallelisation of the simplex method // Computational Management Science. 2010. **7**, N 2. 139–170. doi 10.1007/s10287-008-0080-5.
22. *Mamalis B., Pantziou G.* Advances in the parallelization of the simplex method // Lecture Notes in Computer Science. Vol. 9295. Cham: Springer, 2015. 281–307. doi 10.1007/978-3-319-24024-4_17.
23. *Зоркальцев В.И., Мокрый И.В.* Алгоритмы внутренних точек в линейной оптимизации // Сибирский журнал индустриальной математики. 2018. **21**, № 1. 11–20. doi 10.17377/sibjim.2018.21.102.
24. *Дикин И.И.* Итеративное решение задач линейного и квадратичного программирования // Докл. АН СССР. 1967. **174**, № 4. 747–748. https://www.mathnet.ru/rus/dan33112. Дата обращения: 12 ноября 2023.

2023, **24** (4), 408–429. doi 10.26089/NumMet.v24r428

25. *Gondzio J.* Interior point methods 25 years later // European Journal of Operational Research. 2012. **218**, N 3. 587–601. doi 10.1016/j.ejor.2011.09.017.
26. *Roos C., Terlaky T., Vial J.-P.* Interior point methods for linear optimization. New York: Springer, 2005. doi 10.1007/b100325.
27. *Sokolinskaya I.* Parallel method of pseudoprojection for linear inequalities // Communications in Computer and Information Science. Vol. 910. Cham: Springer, 2018. 216–231. doi 10.1007/978-3-319-99673-8_16.
28. *Gondzio J., Grothey A.* Direct solution of linear systems of size 10^9 arising in optimization with interior point methods // Lecture Notes in Computer Science. Vol. 3911. Berlin: Springer, 2006. 513–525. doi 10.1007/11752578_62.
29. *Prieto A., Prieto B., Ortigosa E.M., et al.* Neural networks: an overview of early research, current frameworks and new challenges // Neurocomputing. 2016. **214**. 242–268. doi 10.1016/j.neucom.2016.06.014.
30. *Tank D., Hopfield J.* Simple 'neural' optimization networks: an A/D converter, signal decision circuit, and a linear programming circuit // IEEE Transactions on Circuits and Systems. 1986. **33**, N 5. 533–541. doi 10.1109/TCS.1986.1085953.
31. *Kennedy M., Chua L.* 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. **34**, N 2. 210–214. doi 10.1109/TCS.1987.1086095.
32. *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. **37**, N 3. 384–398. doi 10.1109/31.52732.
33. *Zak S.H., Upatising V., Hui S.* Solving linear programming problems with neural networks: a comparative study // IEEE Transactions on Neural Networks. 1995. **6**, N 1. 94–104. doi 10.1109/72.363446.
34. *Malek A., Yari A.* Primal–dual solution for the linear programming problems using neural networks // Applied Mathematics and Computation. 2005. **167**, N 1. 198–211. doi 10.1016/J.AMC.2004.06.081.
35. *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. **87**. 39–46. doi 10.1016/j.chaos.2016.03.009.
36. *LeCun Y., Bengio Y., Hinton G.* Deep learning // Nature. 2015. **521**, N 7553. 436–444. doi 10.1038/nature14539.
37. *Ольховский Н.А., Соколинский Л.Б.* Визуальное представление многомерных задач линейного программирования // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2022. **11**, № 1. 31–56. doi 10.14529/cmse220103.
38. *Raina R., Madhavan A., Ng A.Y.* Large-scale deep unsupervised learning using graphics processors // Proc. 26th Annual Int. Conf. on Machine Learning, Montreal, Canada, June 14–18, 2009. New York: ACM Press, 2009. 873–880. doi 10.1145/1553374.1553486.
39. *Соколинский Л.Б., Соколинская И.М.* О новой версии апекс-метода для решения задач линейного программирования // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2023. **12**, № 2. 5–46. doi 10.14529/cmse230201.
40. *Ольховский Н.А.* Исследование структуры рецептивного поля в визуальном методе решения задачи линейного программирования // NPG Publishing. 2023. doi 10.24108/preprints-3112771.

Поступила в редакцию
1 августа 2023 г.

Принята к публикации
7 ноября 2023 г.

Информация об авторах

Николай Александрович Ольховский — аспирант; Южно-Уральский государственный университет (национальный исследовательский университет), пр-кт им. В. И. Ленина, д. 76, 454080, Челябинск, Российская Федерация.

Леонид Борисович Соколинский — д.ф.-м.н., заведующий кафедрой системного программирования; Южно-Уральский государственный университет (национальный исследовательский университет), пр-кт им. В. И. Ленина, д. 76, 454080, Челябинск, Российская Федерация.

2023, **24** (4), 408–429. doi 10.26089/NumMet.v24r428

References

1. T. Hartung, "Making Big Sense from Big Data," Front. Big Data **1**, Article Number 5 (2018). doi 10.3389/fdata.2018.00005.
2. I. M. Sokolinskaya and L. B. Sokolinsky, "On the Solution of Linear Programming Problems in the Age of Big Data," in *Communications in Computer and Information Science* (Springer, Cham, 2017), Vol. 753, pp. 86–100. doi 10.1007/978-3-319-67035-5_7.
3. J. Branke, "Optimization in Dynamic Environments," in *Evolutionary Optimization in Dynamic Environments. Genetic Algorithms and Evolutionary Computation* (Springer, Boston, 2002), Vol. 3, pp. 13–29. doi 10.1007/978-1-4615-0911-0_2.
4. I. I. Eremin and V. D. Mazurov, *Nonstationary Processes of Mathematical Programming* (Nauka, Moscow, 1979) [in Russian].
5. J. Brogaard, T. Hendershott, and R. Riordan, "High-Frequency Trading and Price Discovery," Rev. Financ. Stud. **27** (8), 2267–2306 (2014). doi 10.1093/rfs/hhu032.
6. S. Deng, X. Huang, J. Wang, et al., "A Decision Support System for Trading in Apple Futures Market Using Predictions Fusion," IEEE Access **9**, 1271–1285 (2021). doi 10.1109/ACCESS.2020.3047138.
7. G. Seregin, *Lecture Notes on Regularity Theory for the Navier–Stokes Equations* (World Scientific, Singapore, 2014). doi 10.1142/9314.
8. D. Demin, "Synthesis of Optimal Control of Technological Processes Based on a Multialternative Parametric Description of the Final State," East.-Eur. J. Enterp. Technol. **3** (4), 51–63 (2017).
9. L. S. Kazarinov, D. A. Shnayder, and O. V. Kolesnikova, "Heat Load Control in Steam Boilers," in *Proc. Int. Conf. on Industrial Engineering, Applications and Manufacturing, St. Petersburg, Russia, May 16–19, 2017.* doi 10.1109/ICIEAM.2017.8076177.
10. E. V. Zagoskina, T. A. Barbasova, and D. A. Shnaider, "Intelligent Control System of Blast-Furnace Melting Efficiency," in *Proc. Int. Multi-Conference on Engineering, Computer and Information Sciences, Novosibirsk, Russia, October 21–27, 2019.* doi 10.1109/SIBIRCON48586.2019.8958221.
11. J. Fleming, X. Yan, C. Allison, et al., "Real-Time Predictive Eco-Driving Assistance Considering Road Geometry and Long-Range Radar Measurements," IET Intell. Transp. Syst. **15** (4), 573–583 (2021). doi 10.1049/ITR2.12047.
12. M. Scholl, K. Minnerup, C. Reiter, et al., "Optimization of a Thermal Management System for Battery Electric Vehicles," in *Proc. 14th Int. Conf. on Ecological Vehicles and Renewable Energies, Monte-Carlo, Monaco, May 8–10, 2019.* doi 10.1109/EVER.2019.8813657.
13. S. Meisel, "Dynamic Vehicle Routing," in *Anticipatory Optimization for Dynamic Decision Making. Operations Research/Computer Science Interfaces Series* (Springer, New York, 2011), Vol. 51, pp. 77–96. doi 10.1007/978-1-4614-0505-4_6.
14. D. R. Kiran, *Production Planning and Control: A Comprehensive Approach* (Butterworth-Heinemann, Oxford, 2019). doi 10.1016/C2018-0-03856-6.
15. R. Mall, *Real-Time Systems: Theory and Practice* (Pearson Education, Delhi, 2007).
16. G. B. Dantzig, *Linear Programming and Extensions* (Princeton Univ. Press, Princeton, 1998).
17. J. A. J. Hall and K. I. M. McKinnon, "Hyper-Sparsity in the Revised Simplex Method and How to Exploit It," Comput. Optim. Appl. **32** (3), 259–283 (2005). doi 10.1007/s10589-005-4802-0.
18. V. Klee and G. Minty, "How Good is the Simplex Algorithm?" in *Inequalities III* (Academic Press, New York, 1972), pp. 159–175.
19. R. H. Bartels, J. Stoer, and Ch. Zenger, "A Realization of the Simplex Method Based on Triangular Decompositions," in *Handbook for Automatic Computation. Vol. II: Linear Algebra* (Springer, Berlin, 1971), pp. 152–190. doi 10.1007/978-3-642-86940-2_11.
20. P. Tolla, "A Survey of Some Linear Programming Methods," in *Concepts of Combinatorial Optimization* (Wiley, Hoboken, 2014), pp. 157–188. doi 10.1002/9781119005216.ch7.
21. J. A. J. Hall, "Towards a Practical Parallelisation of the Simplex Method," Comput. Manag. Sci. **7** (2), 139–170 (2010). doi 10.1007/s10287-008-0080-5.
22. B. Mamalis and G. Pantziou, "Advances in the Parallelization of the Simplex Method," in *Lecture Notes in Computer Science* (Springer, Cham, 2015), Vol. 9295, pp. 281–307. doi 10.1007/978-3-319-24024-4_17.
23. V. I. Zorkaltsev and I. V. Mokryi, "Interior Point Algorithms in Linear Optimization," Sib. Zh. Ind. Mat. **21** (1), 11–20 (2018) [J. Appl. Ind. Math. **12** (1), 191–199 (2018)]. doi 10.1134/S1990478918010179.
24. I. I. Dikin, "Iterative Solution of Problems of Linear and Quadratic Programming," Dokl. Akad. Nauk SSSR **174** (4), 747–748 (1967).

2023, **24** (4), 408–429. doi 10.26089/NumMet.v24r428

25. J. Gondzio, "Interior Point Methods 25 Years Later," Eur. J. Oper. Res. **218** (3), 587–601 (2012). doi 10.1016/j.ejor.2011.09.017.
26. C. Roos, T. Terlaky, and J.-P. Vial, *Interior Point Methods for Linear Optimization* (Springer, New York, 2005). doi 10.1007/b100325.
27. I. Sokolinskaya, "Parallel Method of Pseudoprojection for Linear Inequalities," in *Communications in Computer and Information Science* (Springer, Cham, 2018), Vol. 910, pp. 216–231. doi 10.1007/978-3-319-99673-8_16.
28. J. Gondzio and A. Grothey, "Direct Solution of Linear Systems of Size 10^9 Arising in Optimization with Interior Point Methods," in *Lecture Notes in Computer Science* (Springer, Berlin, 2006), Vol. 3911, pp. 513–525. doi 10.1007/11752578_62.
29. A. Prieto, B. Prieto, E. M. Ortigosa, et al., "Neural Networks: An Overview of Early Research, Current Frameworks and New Challenges," Neurocomputing **214**, 242–268 (2016). doi 10.1016/j.neucom.2016.06.014.
30. D. Tank and J. Hopfield, "Simple 'Neural' Optimization Networks: An A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit," IEEE Trans. Circuits Syst. **33** (5), 533–541 (1986). doi 10.1109/TCS.1986.1085953.
31. M. Kennedy and L. Chua, "Unifying the Tank and Hopfield Linear Programming Circuit and the Canonical Nonlinear Programming Circuit of Chua and Lin," IEEE Trans. Circuits Syst. **34** (2), 210–214 (1987). doi 10.1109/TCS.1987.1086095.
32. A. Rodriguez-Vazquez, R. Dominguez-Castro, A. Rueda, et al., "Nonlinear Switched-Capacitor 'Neural' Networks for Optimization Problems," IEEE Trans. Circuits Syst. **37** (3), 384–398 (1990). doi 10.1109/31.52732.
33. S. H. Zak, V. Upatising, and S. Hui, "Solving Linear Programming Problems with Neural Networks: A Comparative Study," IEEE Trans. Neural Netw. **6** (1), 94–104 (1995). doi 10.1109/72.363446.
34. A. Malek and A. Yari, "Primal–Dual Solution for the Linear Programming Problems Using Neural Networks," Appl. Math. Comput. **167** (1), 198–211 (2005). doi 10.1016/J.AMC.2004.06.081.
35. X. Liu and M. Zhou, "A One-Layer Recurrent Neural Network for Non-Smooth Convex Optimization Subject to Linear Inequality Constraints," Chaos Solit. Fractals **87**, 39–46 (2016). doi 10.1016/j.chaos.2016.03.009.
36. Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," Nature **521** (7553), 436–444 (2015). doi 10.1038/nature14539.
37. N. A. Olkhovsky and L. B. Sokolinsky, "Visualizing Multidimensional Linear Programming Problems," in *Communications in Computer and Information Science* (Springer, Cham, 2022), Vol. 1618, pp. 172–196. doi 10.1007/978-3-031-11623-0_13.
38. R. Raina, A. Madhavan, and A. Y. Ng, "Large-Scale Deep Unsupervised Learning Using Graphics Processors," in *Proc. 26th Annual Int. Conf. on Machine Learning, Montreal, Canada, June 14–18, 2009* (ACM Press, New York, 2009), pp. 873–880. doi 10.1145/1553374.1553486.
39. L. B. Sokolinsky and I. M. Sokolinskaya, "Apex Method: A New Scalable Iterative Method for Linear Programming," Mathematics **11** (7), 1654-1–1654-28 (2023). doi 10.3390/MATH11071654.
40. N. A. Olkhovsky, *Study of the Receptive Field Structure in the Visual Method of Solving the Linear Programming Problem* [in Russian]. doi 10.24108/preprints-3112771.

Received
August 1, 2023

Accepted for publication
November 7, 2023

Information about the authors

Nikolay A. Olkhovsky — PhD student; South Ural State University (National Research University), Lenin prospekt 76, 454080, Chelyabinsk, Russia.

Leonid B. Sokolinsky — Dr. Sci., Head of Computer Science Department; South Ural State University (National Research University), Lenin prospekt 76, 454080, Chelyabinsk, Russia.