УДК 519.852, 514.172.45, 519.168 **DOI: 10.14529/cmse250301**

О ВЫЧИСЛЕНИИ ВЕРШИНЫ МНОГОГРАННИКА ДОПУСТИМЫХ РЕШЕНИЙ СИСТЕМЫ ЛИНЕЙНЫХ ОГРАНИЧЕНИЙ

© 2025 А.Э. Жулев, Л.Б. Соколинский, И.М. Соколинская

Южно-Уральский государственный университет

(454080 Челябинск, пр. им. В.И. Ленина, д. 76)

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

Поступила в редакцию: 14.06.2025

В статье предлагается новый алгоритм вычисления вершины многогранника, представляющего собой область допустимых решений системы линейных ограничений. Алгоритм, получивший название VeSP, начинает работу в произвольной точке многогранника и, перемещаясь по его граням, завершает свою работу в некоторой его вершине. Для вычисления направления движения по грани используется проекционный метод, суть которого заключается в следующем. Для точки текущего приближения вычисляется аффинное подпространство, являющееся аффинной оболочкой грани, на которой расположена эта точка. К точке текущего приближения прибавляется ненулевой вектор, дающий «внешнюю» точку. Из «внешней» точки по известной аналитической формуле вычисляется ортогональная проекция на текущее аффинное подпространство. Точка проекции определяет направление движения по грани до ее границы, что дает следующее приближение. При каждом таком перемещении размерность текущей грани уменьшается. В конечном итоге приходим к грани нулевой размерности, то есть к вершине многогранника. Дается формальное описание алгоритма VeSP. Доказывается сходимость алгоритма VeSP к вершине многогранника за конечное число итераций, не превышающих размерность пространства. Приводится информация о реализации алгоритма VaSP на языке C++. Описываются результаты вычислительных экспериментов на эталонных задачах из репозитория Netlib-LP. Результаты экспериментов показывают, что алгоритм VeSP применительно ко всем тестовым задачам успешно нашел вершину многогранника за количество итераций, не превышающее размерность пространства. Для большинства задач время нахождения вершины на обычном персональном компьютере потребовало менее одной секунды.

Ключевые слова: линейные ограничения, многогранник допустимых решений, вычисление вершины, проекционный метод, алгоритм VeSP.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Жулев А.Э., Соколинский Л.Б., Соколинская И.М. О вычислении вершины многогранника допустимых решений системы линейных ограничений // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2025. Т. 14, № 3. С. 5–27. DOI: 10.14529/cmse250301.

Введение

Задача нахождения вершины многогранника, задаваемого системой линейных неравенств и уравнений, встречается во всех областях, использующих линейное программирование (ЛП) в качестве основной оптимизационной модели. Так, симплекс-метод [1] для своей работы требует некоторую стартовую вершину многогранника допустимых решений. Проекционный алгоритм AlEM решения задачи ЛП, предложенный в недавней работе [2], также требует в качестве стартовой точки вершину многогранника допустимых решений. Стартовая вершина также необходима в алгоритмах перечисления всех вершин многогранника (см., например, [3, 4]), используемых в таких областях, как теория игр, исследование операций, математическая биология и кристаллография.

Рассмотрим систему линейных ограничений стандартного вида[1]

$$A\boldsymbol{x} \leqslant \boldsymbol{b}, \tag{1}$$

где $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ — вектор переменных, A — вещественная матрица размера $m \times n$, $\boldsymbol{b}$ — столбец вещественных чисел размера m. Необходимо найти вершину многогранника $M \subset \mathbb{R}^n$, представляющего собой область допустимых решений системы (1). Предполагается, что многогранник M является ограниченным множеством. Наиболее известными методами вычисления вершины многогранника M являются симплекс-метод и метод исключения переменных Фурье—Моцкина.

Идея подхода, основанного на симплекс-методе, заключается в следующем [5]. Переупорядочив строки системы (1), представим ее в виде

$$\begin{cases} A^+\boldsymbol{x} \leqslant \boldsymbol{b}^+ \\ A^-\boldsymbol{x} \geqslant \boldsymbol{b}^-, \end{cases}$$

где $\boldsymbol{b}^+ \geqslant 0$, $\boldsymbol{b}^- > 0$. Рассмотрим задачу ЛП следующего вида[2]:

$$\max\{\langle \mathbf{1}, A^+\boldsymbol{x} - \boldsymbol{z}\rangle \mid \boldsymbol{z} \geqslant 0;\ A^+\boldsymbol{x} \leqslant \boldsymbol{b}^+;\ A^-\boldsymbol{x} - \boldsymbol{z} \leqslant \boldsymbol{b}^-\}, \tag{2}$$

где $\boldsymbol{z} = (z_1, \ldots, z_n) \in \mathbb{R}^n$ — новый вектор переменных, и $\mathbf{1} = (1, \ldots 1) \in \mathbb{R}^n$ — вектор единиц. В этом случае $\boldsymbol{x} = \mathbf{0}$ и $\boldsymbol{z} = \mathbf{0}$ задают вершину многогранника допустимых решений задачи (2). Таким образом, мы можем решить задачу (2) с помощью симплекс-метода. Если максимальное значение задачи (2) равно $\langle \mathbf{1}, \boldsymbol{b}^-\rangle$, и достигается в вершине $(\boldsymbol{x}^*, \boldsymbol{y}^*)$, то $\boldsymbol{x}^*$ будет вершиной многогранника M, представляющего область допустимых решений системы (1). Если максимальное значение задачи (2) меньше $\langle \mathbf{1}, \boldsymbol{b}^-\rangle$, то система (1) не имеет решений. Главным недостатком этого подхода является то, симплекс-метод в общем случае может потребовать экспоненциальное число шагов для получения решения [6].

Метод Фурье—Моцкина [7] основан на решении системы линейных неравенств путем последовательного исключения переменных подобно тому, как это делается при решении системы линейных уравнений. Впервые метод исключения переменных применительно к линейным неравенствам был упомянут Фурье в 1827 году (см. [8]). Моцкин повторно открыл и использовал этот метод в своей докторской диссертации [9]. Следуя [5], рассмотрим метод Фурье—Моцкина применительно к задаче вычисления вершины многогранника допустимых решений системы (1). Обозначим a_i^j — j-тый элемент i-той строки матрицы A. Путем умножения неравенств на соответствующие положительные числа, приведем систему (1) к виду

$$\begin{cases} x_1 + \langle \boldsymbol{a}_i^+, \boldsymbol{x}'\rangle \leqslant b_i^+ & (i = 1, \ldots, m') \\ -x_1 + \langle \boldsymbol{a}_i^-, \boldsymbol{x}'\rangle \leqslant b_i^- & (i = m' + 1, \ldots, m'') \\ \langle \boldsymbol{a}_i^0, \boldsymbol{x}'\rangle \leqslant b_i & (i = m'' + 1, \ldots, m), \end{cases} \tag{3}$$

[1]Заметим, что система линейных ограничений, включающая равенства, может быть легко приведена к стандартному виду путем замены каждого равенства парой противоположных нестрогих неравенств. В свою очередь неравенство «больше, либо равно» может быть преобразовано в неравенство «меньше, либо равно» путем умножения обеих частей на -1. Мы здесь исключаем системы линейных ограничений, включающие строгие неравенства, так как это влечет открытость области допустимых решений.

[2]Здесь $\langle \cdot\ ,\ \cdot \rangle$ обозначает скалярное произведение векторов.

где $\boldsymbol{a}_i^+ = \left(a_i^2/a_i^1, \ldots, a_i^n/a_i^1\right)$, $\boldsymbol{a}_i^- = \left(-a_i^2/a_i^1, \ldots, -a_i^n/a_i^1\right)$, $\boldsymbol{a}_i^0 = \left(a_i^2, \ldots, a_i^n\right)$, $b_i^+ = b_i/a_i^1$, $b_i^- = -b_i/a_i^1$, $\boldsymbol{x}' = (x_2, \ldots, x_n)$. Поскольку первые две строки системы (3) равносильны условию

$$\max_{i\in\{m'+1,\ldots,m''\}} \left(\langle \boldsymbol{a}_i^-, \boldsymbol{x}'\rangle - b_i^-\right) \leqslant x_1 \leqslant \min_{i\in\{1,\ldots,m'\}} \left(b_i^+ - \langle \boldsymbol{a}_i^+, \boldsymbol{x}'\rangle\right), \tag{4}$$

переменная x_1 может быть исключена. Это позволяет перейти к следующей эквивалентной системе:

$$\begin{cases} \langle \boldsymbol{a}_i^-, \boldsymbol{x}'\rangle - b_i^- \leqslant b_1^+ - \langle \boldsymbol{a}_1^+, \boldsymbol{x}'\rangle & (i = m'+1, \ldots, m'') \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots & \ldots\ldots\ldots\ldots\ldots \\ \langle \boldsymbol{a}_i^-, \boldsymbol{x}'\rangle - b_i^- \leqslant b_{m'}^+ - \langle \boldsymbol{a}_{m'}^+, \boldsymbol{x}'\rangle & (i = m'+1, \ldots, m'') \\ \langle \boldsymbol{a}_i^0, \boldsymbol{x}'\rangle \leqslant b_i & (i = m''+1, \ldots, m), \end{cases}$$

которая равносильна системе

$$\begin{cases} \langle \boldsymbol{a}_i^- + \boldsymbol{a}_1^+, \boldsymbol{x}'\rangle \leqslant b_1^+ + b_i^- & (i = m'+1, \ldots, m'') \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots & \ldots\ldots\ldots\ldots\ldots \\ \langle \boldsymbol{a}_i^- + \boldsymbol{a}_{m'}^+, \boldsymbol{x}'\rangle \leqslant b_{m'}^+ + b_i^- & (i = m'+1, \ldots, m'') \\ \langle \boldsymbol{a}_i^0, \boldsymbol{x}'\rangle \leqslant b_i & (i = m''+1, \ldots, m). \end{cases} \tag{5}$$

Эта система включает в себя $m'(m'' - m') + m - m''$ ограничений и $n - 1$ переменных. В действительности, система (5) определяет проекцию многогранника допустимых решений M исходной системы (1) на гиперплоскость $x_1 = 0$. Проекция многогранника является многогранником. Обозначим получившийся многогранник через M_1. Если размерность многогранника M равна n, то размерность многогранника M_1 будет равна $n - 1$:

$$\dim(M_1) = n - 1.$$

Продолжая процесс исключения переменных выше описанным способом, в итоге получим многогранник M_{n-1} размерности 1, являющийся проекцией многогранника M_{n-2} на гиперплоскость $x_n = 0$ и представляющий собой отрезок. Очевидно, что в любой точке этого отрезка переменные $x_1, \ldots, x_{n-1}$ принимают значение 0, а значения переменной x_n ограничиваются условием

$$\beta^- \leqslant x_n \leqslant \beta^+,$$

где β^-, β^+ — скалярные величины, вычисленные в ходе исключения переменных. При этом вершинам многогранника M_{n-1} соответствуют минимальное и максимальное допустимые значения переменной x_n. Для определенности положим $x_n^* = \beta^+$ — n-ная координата вершины многогранника. Двигаясь в обратном порядке, вычислим остальные координаты искомой вершины, используя ограничения вида (4). Например, если мы уже нашли координаты $x_2^*, \ldots, x_n^*$, подставив их в правую часть (4) в качестве $\boldsymbol{x}'$, получим

$$x_1^* = \min_{i\in\{1,\ldots,m'\}} \left(b_i^+ - \langle \boldsymbol{a}_i^+, (x_2^*, \ldots, x_n^*)\rangle\right).$$

Таким образом мы вычислим вершину $\boldsymbol{x}^* = (x_1^*, \ldots, x_n^*)$ многогранника допустимых решений M для исходной системы ограничений (1). Описанный метод имеет тот же недостаток, что и симплекс-метод: в общем случае метод исключения Фурье—Моцкина характеризуется экспоненциальной временной сложностью [10].

В этой статье мы предлагаем новый метод вычисления вершины многогранника допустимых решений системы линейных неравенств, который позволяет вычислить вершину не более, чем за n итераций, где n — размерность пространства. Проекционный алгоритм, реализующий этот метод, получил название VeSP (Vertex Seeking by Projecting). Статья организована следующим образом. Раздел 1 содержит обозначения, определения и утверждения, необходимые для описания алгоритма VeSP и его численной реализации. В разделе 2 дается формализованное описание алгоритма VeSP, и доказывается утверждение о сходимости этого алгоритма. В разделе 3 представлены информация о программной реализации алгоритма VeSP и результаты вычислительных экспериментов. В заключении суммируются полученные результаты и намечаются направления дальнейших исследований. В конце статьи приводится сводка используемых математических обозначений.

1. Определения и обозначения

В этом разделе приводятся основные определения и обозначения, используемые для описания алгоритма VeSP.

В евклидовом пространстве $\mathbb{R}^n$ рассматривается система линейных ограничений общего вида, включающая в себя m линейных неравенств и k линейных уравнений:

$$\begin{cases} \hat{A}\boldsymbol{x} \leqslant \hat{\boldsymbol{b}}; \\ \bar{A}\boldsymbol{x} = \bar{\boldsymbol{b}}, \end{cases} \tag{6}$$

где $\hat{A} \in \mathbb{R}^{m\times n}$, $\bar{A} \in \mathbb{R}^{k\times n}$, $\hat{\boldsymbol{b}} \in \mathbb{R}^m$ и $\bar{\boldsymbol{b}} \in \mathbb{R}^k$. Предполагается, что размерность пространства $n > 1$, количество уравнений $k \geqslant 0$. Матрицу $\hat{A}$ будем называть граничной, а матрицу $\bar{A}$ — опорной. Предполагается, что система (6) включает в себя также неравенства вида

$$\begin{aligned} -x_1 &\leqslant 0; \\ &\cdots\cdots\cdots \\ -x_n &\leqslant 0. \end{aligned} \tag{7}$$

Таким образом, число неравенств m не может быть меньше размерности пространства n:

$$m \geqslant n. \tag{8}$$

Из (7) и (8) следует

$$\operatorname{rank}\left(\hat{A}\right) = n. \tag{9}$$

Подсистема

$$\hat{A}\boldsymbol{x} \leqslant \hat{\boldsymbol{b}}$$

задает m замкнутых полупространств[3]

$$\begin{aligned} P_1 &= \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_1, \boldsymbol{x}\rangle \leqslant b_1\}; \\ &\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ P_m &= \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_m, \boldsymbol{x}\rangle \leqslant b_m\}. \end{aligned} \tag{10}$$

[3]Здесь и далее мы опускаем прилагательное «аффинных» в отношении полупространств, гиперплоскостей и подпространств.

Здесь $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m$ обозначают строки матрицы $\hat{A}$; $b_1, \ldots, b_m$ — элементы столбца $\hat{\boldsymbol{b}}$. Пересечение полупространств (10) образует замкнутый выпуклый многогранник $\hat{M}$, называемый граничным:

$$\hat{M} = \bigcap_{i=1}^{m} P_i. \tag{11}$$

Полупространства (10) ограничиваются гиперплоскостями, называемыми граничными:

$$\begin{gathered} \hat{H}_1 = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_1, \boldsymbol{x} \rangle = b_1\}; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \hat{H}_m = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_m, \boldsymbol{x} \rangle = b_m\}. \end{gathered} \tag{12}$$

Без ограничения общности мы можем предполагать, что среди граничных гиперплоскостей нет совпадающих.

Подсистема

$$\bar{A}\boldsymbol{x} = \bar{\boldsymbol{b}}$$

задает k гиперплоскостей, называемых опорными:

$$\begin{gathered} \bar{H}_{m+1} = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_{m+1}, \boldsymbol{x} \rangle = b_{m+1}\}; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \bar{H}_{m+k} = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_{m+k}, \boldsymbol{x} \rangle = b_{m+k}\}. \end{gathered} \tag{13}$$

Здесь $\boldsymbol{a}_{m+1}, \ldots, \boldsymbol{a}_{m+k}$ обозначают строки матрицы $\bar{A}$; $b_{m+1}, \ldots, b_{m+k}$ — элементы столбца $\bar{\boldsymbol{b}}$. Пересечение опорных гиперплоскостей образует опорное подпространство $\bar{S}$:

$$\bar{S} = \begin{cases} \bigcap\limits_{i=1}^{k} \bar{H}_{m+i}, & \text{если} \quad k > 0; \\ \mathbb{R}^n, & \text{если} \quad k = 0. \end{cases} \tag{14}$$

Область допустимых решений, определяемая системой ограничений (6), представляет собой замкнутый выпуклый многогранник M, называемый допустимым:

$$M = \bar{S} \cap \hat{M}. \tag{15}$$

В частности, из (15) следуют

$$M \subseteq \bar{S}; \tag{16}$$

$$M \subseteq \hat{M}. \tag{17}$$

Мы будем предполагать, что допустимый многогранник M является непустым ограниченным множеством.

Обозначим через $\hat{I}$ множество индексов строк граничной матрицы $\hat{A}$, и через $\bar{I}$ — множество индексов строк опорной матрицы $\bar{A}$:

$$\hat{I} = \{1, \ldots, m\}; \tag{18}$$

$$\bar{I} = \{m+1, \ldots, m+k\}. \tag{19}$$

В соответствии с (19) формулу (14) можно переписать в виде

$$\bar{S} = \begin{cases} \bigcap\limits_{i \in \bar{I}} \bar{H}_i\,, & \text{если} \quad \bar{I} \neq \emptyset; \\ \mathbb{R}^n\,, & \text{если} \quad \bar{I} = \emptyset. \end{cases} \tag{20}$$

Определение 1. Систему ограничений (6) будем называть регулярной, если выполняются следующие условия:

$$k < n; \tag{21}$$

$$\operatorname{rank}\left(\bar{A}\right) = k; \tag{22}$$

$$\dim(M) = n - k; \tag{23}$$

$$\forall \boldsymbol{v} \in M \;:\; \operatorname{rank}\left(\begin{bmatrix} \bar{A} \\ \hat{A}_{\boldsymbol{v}} \end{bmatrix}\right) = \min(k + l, n), \tag{24}$$

где l — количество граничных гиперплоскостей, проходящих через точку $\boldsymbol{v}$, $\hat{A}_{\boldsymbol{v}}$ — матрица размера $l \times n$, строками которой являются коэффициенты уравнений граничных гиперплоскостей, проходящих через точку $\boldsymbol{v}$.

Условие (21) означает, что количество уравнений в системе ограничений должно быть меньше размерности пространства (в противном случае допустимый многогранник M вырождается в точку). Условие (22) говорит, что опорная матрица $\bar{A}$ должна иметь полный ранг, то есть среди ограничений не должно быть уравнений, задающих различные параллельные гиперплоскости. Условие (23) требует, чтобы допустимый многогранник M имел размерность, равную $n - k$. Это означает, что в системе ограничений не может быть неявных равенств[4]. Согласно условию (24) в системе ограничений не должно быть избыточных равенств, и при этом ранг матрицы коэффициентов неравенств, содержащих граничную точку, должен быть равным минимуму от числа неравенств и размерности пространства. Заметим, что для любой системы линейных ограничений, областью допустимых решений которой является ограниченное множество размерности больше 0, существует ее эквивалентное представление с регулярной системой ограничений в пространстве той же размерности (см. [11], теорема 2.15).

Везде далее будем предполагать, что система ограничений (6) является регулярной. Это обеспечивает корректность всех последующих утверждений и алгоритмов. Следующее утверждение является теоретической основой для построения алгоритма VeSP.

Утверждение 1. Пусть выполняются условия (21)–(24). Пусть через точку $\boldsymbol{v} \in M$ проходит l граничных гиперплоскостей. Обозначим через $\hat{I}_{\boldsymbol{v}}$ множество индексов этих граничных гиперплоскостей:

$$\boldsymbol{v} \in \bigcap_{i \in \hat{I}_{\boldsymbol{v}}} \hat{H}_i. \tag{25}$$

[4]Неравенство $\langle \boldsymbol{a}, \boldsymbol{x} \rangle \leqslant b$ из $A\boldsymbol{x} \leqslant \boldsymbol{b}$ является неявным равенством, если $\langle \boldsymbol{a}, \boldsymbol{x} \rangle = b$ для всех $\boldsymbol{x}$, удовлетворяющих $A\boldsymbol{x} \leqslant \boldsymbol{b}$.

Положим

$$S_{\boldsymbol{v}} = \begin{cases} \left(\bigcap\limits_{i \in \hat{I}_v} \hat{H}_i \right) \cap \bar{S}\,, & \text{если} \quad \hat{I}_{\boldsymbol{v}} \neq \emptyset; \\ \mathbb{R}^n\,, & \text{если} \quad \hat{I}_{\boldsymbol{v}} = \emptyset; \end{cases} \tag{26}$$

$$F = M \cap S_{\boldsymbol{v}}. \tag{27}$$

Тогда множество F является гранью допустимого многогранника M и

$$\dim(F) = n - \min(k + l, n). \tag{28}$$

Доказательство. Сначала предположим, что $l = 0$, то есть через точку $\boldsymbol{v}$ не проходит ни одной граничной гиперплоскости. Следовательно, $\hat{I}_{\boldsymbol{v}} = \emptyset$. Тогда в силу (20) и (26) имеем

$$S_{\boldsymbol{v}} = \bar{S}.$$

С учетом (16) отсюда следует

$$M \subseteq S_{\boldsymbol{v}}.$$

Принимая во внимание (27), таким образом получаем

$$F = M. \tag{29}$$

Сам многогранник M является своей гранью [11]. Так как система ограничений (6) по предположению является регулярной, из (23) и (29) следует $\dim(F) = n - k$. Таким образом, для $l = 0$ утверждение доказано.

Пусть теперь $l > 0$, то есть $\hat{I}_{\boldsymbol{v}} \neq \emptyset$. В силу (11), (17) и (25) для любого $i \in \hat{I}_{\boldsymbol{v}}$ имеем

$$M \subseteq P_i;$$
$$M \cap \hat{H}_i \neq \emptyset.$$

Это — достаточные условия для того, чтобы множество $F_i = M \cap \hat{H}_i$ являлось гранью выпуклого многогранника M (см. [11], определение 2.1). Пересечение граней многогранника M — снова грань многогранника M (см. [11], утверждение 2.3). Поэтому множество

$$G = M \cap \left(\bigcap_{i \in \hat{I}_v} \hat{H}_i \right) \tag{30}$$

является гранью многогранника M. Покажем, что $F = G$. Сначала предположим, что $\bar{I} = \emptyset$. Тогда из (26) следует

$$S_{\boldsymbol{v}} = \bigcap_{i \in \hat{I}_v} \hat{H}_i.$$

С учетом (27) и (30) отсюда получаем

$$F = M \cap S_{\boldsymbol{v}} = M \cap \left(\bigcap_{i \in \hat{I}_v} \hat{H}_i \right) = M \cap S_{\boldsymbol{v}} = G,$$

то есть F является гранью допустимого многогранника M. Пусть теперь $\bar{I} \neq \emptyset$. Тогда с учетом (16), (20), (26) и (30) имеем

$$G = M \cap \left(\bigcap_{i \in \hat{I}_{\boldsymbol{v}}} \hat{H}_i \right) = M \cap \left(\bigcap_{i \in \bar{I}} \bar{H}_i \right) \cap \left(\bigcap_{i \in \hat{I}_{\boldsymbol{v}}} \hat{H}_i \right) = M \cap \left(\bigcap_{i \in \hat{I}_{\boldsymbol{v}}} \hat{H}_i \right) \cap \bar{S} = M \cap S_{\boldsymbol{v}} = F,$$

то есть и в этом случае F является гранью допустимого многогранника M. Вычислим размерность F для случая $l > 0$. Для этого заметим, что из (22) следует

$$\dim(\bar{S}) = n - k.$$

В силу (16) и (23) отсюда получаем, что подпространство $\bar{S}$ является аффинной оболочкой допустимого многогранника M:

$$\operatorname{aff}(M) = \bar{S}. \tag{31}$$

Из (20) и (26) следует

$$\bar{S} \subseteq S_{\boldsymbol{v}}. \tag{32}$$

Обозначим через $\hat{A}_{\boldsymbol{v}}$ матрицу размера $l \times n$, содержащую коэффициенты граничных уравнений с индексами из $\hat{I}_{\boldsymbol{v}}$. Так как система ограничений (6) по предположению является регулярной, то в соответствии с определением 1 выполняется равенство

$$\operatorname{rank}\left(\begin{bmatrix} \bar{A} \\ \hat{A}_{\boldsymbol{v}} \end{bmatrix} \right) = \min(k + l, n).$$

Принимая во внимание (26), отсюда следует

$$\dim(S_{\boldsymbol{v}}) = n - \min(k + l, n). \tag{33}$$

Учитывая, что аффинная оболочка подпространства есть само это подпространство, и принимая во внимание (27), (31), (32) и (33), имеем

$$\dim(F) = \dim(\operatorname{aff}(F)) = \dim(\operatorname{aff}(M) \cap S_{\boldsymbol{v}}) = \dim(\bar{S} \cap S_{\boldsymbol{v}}) = \dim(S_{\boldsymbol{v}}) = n - \min(k + l, n).$$

Утверждение доказано. □

Определение 2. Собственной гранью точки $\boldsymbol{v} \in M$ будем называть грань наименьшей размерности, содержащую точку $\boldsymbol{v}$.

Утверждение 2. Точка $\boldsymbol{v} \in M$ имеет только одну собственную грань.

Доказательство. Предположим, имеются две различные грани F' и F'' минимальной размерности, содержащие точку $\boldsymbol{v}$. Поскольку пересечение граней допустимого многогранника M является гранью допустимого многогранника M, то $F' \cap F''$ будет гранью допустимого многогранника M, имеющей размерность меньше минимальной, и содержащей точку $\boldsymbol{v}$. Пришли к противоречию. □

Обозначим через face($\boldsymbol{v}$) собственную грань точки $\boldsymbol{v} \in M$.

Утверждение 3. Пусть $\boldsymbol{v} \in M$. Тогда собственная грань точки $\boldsymbol{v}$ может быть вычислена по формуле

$$\text{face}(\boldsymbol{v}) = \begin{cases} M \cap \left(\bigcap\limits_{i \in \hat{I}_{\boldsymbol{v}}} \hat{H}_i \right) \cap \bar{S}\,, & \text{если} \quad \hat{I}_{\boldsymbol{v}} \neq \emptyset; \\ M\,, & \text{если} \quad \hat{I}_{\boldsymbol{v}} = \emptyset, \end{cases} \tag{34}$$

где $\hat{I}_{\boldsymbol{v}}$ — множество индексов всех граничных гиперплоскостей, содержащих точку $\boldsymbol{v}$.

Доказательство. Положим

$$F = \begin{cases} M \cap \left(\bigcap\limits_{i \in \hat{I}_{\boldsymbol{v}}} \hat{H}_i \right) \cap \bar{S}\,, & \text{если} \quad \hat{I}_{\boldsymbol{v}} \neq \emptyset; \\ M\,, & \text{если} \quad \hat{I}_{\boldsymbol{v}} = \emptyset, \end{cases}$$

где $\hat{I}_{\boldsymbol{v}}$ — множество индексов всех граничных гиперплоскостей, содержащих точку $\boldsymbol{v}$. Множество F является гранью допустимого многогранника M в соответствии с утверждением 1. Грань F имеет минимальную размерность среди всех граней, содержащих точку $\boldsymbol{v}$, так как она является подмножеством пересечения всех граничных гиперплоскостей, проходящих через $\boldsymbol{v}$. □

Следствие 1. Пусть $\boldsymbol{v} \in M$. Тогда

$$\dim(\text{face}(\boldsymbol{v})) = n - \min(k + l, n), \tag{35}$$

где l — количество граничных гиперплоскостей, проходящих через точку $\boldsymbol{v}$.

Доказательство. Следует непосредственно из утверждений 1 и 3. □

Следствие 2. Пусть через точку $\boldsymbol{v} \in M$ проходят $n-k$ или более граничных гиперплоскостей. Тогда $\dim(\text{face}(\boldsymbol{v})) = 0$, то есть точка $\boldsymbol{v}$ является вершиной допустимого многогранника M.

Следствие 3. Пусть через точку $\boldsymbol{v} \in M$ проходят $n - k - 1$ граничных гиперплоскостей. Тогда $\dim(\text{face}(\boldsymbol{v})) = 1$, то есть грань face($\boldsymbol{v}$) является ребром допустимого многогранника M.

Следствие 4. Пусть через точку $\boldsymbol{v} \in M$ проходит 0 граничных гиперплоскостей. Тогда $\dim(\text{face}(\boldsymbol{v})) = n - k$ и face($\boldsymbol{v}$) $= M$, то есть сам допустимый многогранник M является собственной гранью точки $\boldsymbol{v}$.

Доказательство. Следует непосредственно из формул (23), (35) и утверждения (2). □

2. Формализованное описание алгоритма VeSP

Этот раздел посвящен формализованному описанию алгоритма VeSP, вычисляющего вершину допустимого многогранника. Алгоритм VeSP начинает свою работу в произвольной точке $\boldsymbol{v}_0$, принадлежащей некоторой грани F_0 допустимого многогранника M. Обозначим через l_0 размерность этой грани. Двигаясь по прямой в произвольном направлении по грани F_0, алгоритм VeSP достигает ее границы в некоторой точке $\boldsymbol{v}_1$. Эта точка будет

находиться на грани F_1, размерность которой l_1 будет меньше l_0. Продолжая движение по граням допустимого многогранника описанным образом, алгоритм VeSP за конечное число шагов придет к точке $\boldsymbol{v}_k$, лежащей на грани F_k, имеющей размерность 0. Точка $\boldsymbol{v}_k$ и будет искомой вершиной допустимого многогранника.

Основной функцией, используемой алгоритмом VeSP, является функция MoveAlongFace$(\cdot)$, выполняющая прохождение грани допустимого многогранника. Эта функция, в свою очередь, использует следующие две функции: $\boldsymbol{\pi}_J(\cdot)$ — вычисление ортогональной проекции на подпространство и Jump$(\cdot)$ — перемещение по направляющему вектору.

Рассмотрим сначала функцию вычисления ортогональной проекции $\boldsymbol{\pi}_J(\boldsymbol{x})$ точки $\boldsymbol{x}$ на подпространство

$$S_J = \bigcap_{i \in J} H_i \quad (J \subseteq \hat{I} \cup \bar{I}), \tag{36}$$

При построении ортогональной проекции мы не различаем граничные и опорные гиперплоскости. Мы также предполагаем, что $S_J \neq \emptyset$. Обозначим через A_J матрицу коэффициентов уравнений с индексами из множества J, через $\boldsymbol{b}_J$ — столбец правых частей этих уравнений. В случае, когда система уравнений $A_J\boldsymbol{x} = \boldsymbol{b}_J$ является совместной, и матрица $A_J A_J^T$ является невырожденной, ортогональная проекция точки $\boldsymbol{x}$ на подпространство S_J может быть вычислена по следующей формуле [12]:

$$\boldsymbol{\pi}_J(\boldsymbol{x}) = \boldsymbol{x} - A_J^T \left(A_J A_J^T\right)^{-1} (A_J\boldsymbol{x} - \boldsymbol{b}_J)\,. \tag{37}$$

Для того, чтобы гарантировать невырожденность матрицы $A_J A_J^T$ в общем случае, необходимо из множества J удалить индексы всех избыточных уравнений[5]. Получившееся множество J' будет обладать следующими свойствами:

$$\begin{aligned} &S_J = S_{J'};\\ &\operatorname{rank}(A_J) = \operatorname{rank}(A_{J'});\\ &\det\left(A_{J'} A_{J'}^T\right) \neq 0. \end{aligned}$$

Если, тем не менее, не удается построить обратную матрицу к матрице $A_{J'} A_{J'}^T$, например, в следствии потери точности, то в этом случае можно приближенно вычислить ортогональную проекцию точки на подпространство, используя алгоритм Качмажа [13, 14].

Далее рассмотрим функцию Jump$(\boldsymbol{x}, \boldsymbol{d})$, осуществляющую перемещение по направляющему вектору $\boldsymbol{d}$ от допустимой точки $\boldsymbol{x} \in M$ к допустимой точке, максимально удаленной от $\boldsymbol{x}$. Основной элементарной операцией, используемой для перемещения, является d-проекция $\boldsymbol{\rho}_i^{\boldsymbol{d}}(\boldsymbol{x})$ точки $\boldsymbol{x}$ на граничную гиперплоскость $\hat{H}_i$, удовлетворяющую условию $\langle \boldsymbol{a}_i, \boldsymbol{d}\rangle > 0$:

$$\boldsymbol{\rho}_i^{\boldsymbol{d}}(\boldsymbol{x}) = \boldsymbol{x} + \gamma, \tag{38}$$

где

$$\gamma = \frac{b_i - \langle \boldsymbol{a}_i, \boldsymbol{x}\rangle}{\langle \boldsymbol{a}_i, \boldsymbol{d}\rangle} \tag{39}$$

(см. определение 1 и утверждение 2 в [15]). Реализация функции Jump$(\boldsymbol{x}, \boldsymbol{d})$ представлена в виде алгоритма 1.

[5]Уравнение с индексом $j \in J$ является избыточным, если $\operatorname{rank}(A_J) = \operatorname{rank}(A_{J-\{j\}})$.

Алгоритм 1 Реализация функции Jump$(\boldsymbol{x}, \boldsymbol{d})$

Require: $\boldsymbol{x} \in M$; $\forall i \in \bar{I} : \langle \boldsymbol{a_i}, \boldsymbol{d} \rangle = 0$

```
1:  function Jump(x, d)
2:      γ_min := +∞
3:      for all i ∈ Î do
4:          if ⟨a_i, d⟩ > 0 then
5:              γ := (b_i − ⟨a_i, x⟩) / ⟨a_i, d⟩
6:              if γ < γ_min then
7:                  γ_min := γ
8:              end if
9:          end if
10:     end for
11:     x_nex = x + γ_min d
12:     return x_nex
13: end function
```

Для работы алгоритма необходимо, чтобы точка $\boldsymbol{x}$ принадлежала допустимому многограннику M, и выполнялось условие

$$\forall i \in \bar{I} : \langle \boldsymbol{a_i}, \boldsymbol{d} \rangle = 0, \tag{40}$$

которое означает, что вектора $\boldsymbol{d}$ должен быть параллелен всем опорным гиперплоскостям (13). Другими словами, точка $(\boldsymbol{x} + \boldsymbol{d})$ обязана удовлетворять всем уравнениям из системы ограничений (6). Схема работы алгоритма 1 проиллюстрирована на рис. 1.

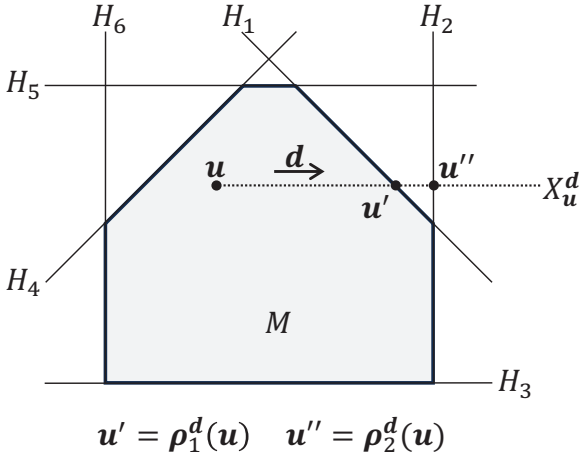

Рис. 1. Действие функции Jump:
Jump$(\boldsymbol{u}, \boldsymbol{d}) = \boldsymbol{u'}$

В приведенном примере изображен многогранник M, образованный пересечением шести полупространств

$$P_1 = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_1, \boldsymbol{x} \rangle \leqslant b_1\};$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$P_6 = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_6, \boldsymbol{x} \rangle \leqslant b_6\}.$$

Эти полупространства ограниченны гиперплоскостями

$$H_1 = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_1, \boldsymbol{x} \rangle = b_1\};$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$H_6 = \{\boldsymbol{x} \in \mathbb{R}^n | \langle \boldsymbol{a}_6, \boldsymbol{x} \rangle = b_6\}.$$

Заданы точка $\boldsymbol{u} \in M$ и направляющий вектор $\boldsymbol{d}$. Необходимо перенести точку $\boldsymbol{u}$ по направлению вектора $\boldsymbol{d}$ в максимально удаленную точку $\boldsymbol{u}' \in M$. Известно [16], что луч

$$X_{\boldsymbol{u}}^{\boldsymbol{d}} = \{\boldsymbol{x} \in \mathbb{R}^n | \, \boldsymbol{x} = \boldsymbol{u} + \lambda \boldsymbol{d}, \, \lambda \geqslant 0\}$$

пересечет гиперплоскость H_i $(i = 1, \ldots, 6)$ только в том случае, когда

$$\langle \boldsymbol{a}_i, \boldsymbol{d} \rangle > 0. \tag{41}$$

Проверка условия (41) на шаге 4 алгоритма 1 в этом случае покажет, что луч $X_{\boldsymbol{u}}^{\boldsymbol{d}}$ пересечет только гиперплоскости H_1 и H_2. Шаги 5–8 алгоритма 1 вычислят d-проекции $\boldsymbol{\rho}_1^{\boldsymbol{d}}(\boldsymbol{u})$ и $\boldsymbol{\rho}_2^{\boldsymbol{d}}(\boldsymbol{u})$ на гиперплоскости H_1, H_2 и выберут ту из них, которая находится ближе всего к точке $\boldsymbol{u}$. Это будет $\boldsymbol{u}' = \boldsymbol{\rho}_1^{\boldsymbol{d}}(\boldsymbol{u})$.

Реализация функции MoveAlongFace($\boldsymbol{v}$), осуществляющей прохождение грани допустимого многогранника, представлена в виде алгоритма 2. Прокомментируем этот алгоритм. Начальная точка $\boldsymbol{v}$ должна принадлежать допустимому многограннику M и не являться его вершиной. На шагах 2–7 строится множество $I_{\boldsymbol{v}}$, включающее в себя индексы всех опорных гиперплоскостей и индексы граничных гиперплоскостей, проходящих через точку $\boldsymbol{v}$. На шагах 8–13 из множества $I_{\boldsymbol{v}}$ удаляются индексы избыточных уравнений в случае, когда они там присутствуют. Если $I_{\boldsymbol{v}} \neq \emptyset$, то далее выполняются шаги 15–19, в результате чего вычисляется точка $\boldsymbol{w} \in \mathrm{aff}(\mathrm{face}(\boldsymbol{v}))$ такая, что $\boldsymbol{w} \neq \boldsymbol{v}$. Для этого на шаге 16 генерируется случайный вектор $\boldsymbol{c} \in \mathbb{R}^n$. Используя этот вектор, шаг 17 вычисляет точку $\boldsymbol{u}$, удаленную от $\boldsymbol{v}$ на расстояние δ. Здесь δ — «большой» положительный параметр: чем больше δ, тем точнее будет вычислен направляющий вектор $\boldsymbol{d}$, используемый функцией Jump$(\cdot)$. Шаг 18 вычисляет точку $\boldsymbol{w}$, являющуюся ортогональной проекцией точки $\boldsymbol{u}$ на подпространство $\mathrm{aff}(\mathrm{face}(\boldsymbol{v}))$ с помощью формулы (37). Если сгенерированный вектор $\boldsymbol{c}$ случайно оказался перпендикулярным к подпространству $\mathrm{aff}(\mathrm{face}(\boldsymbol{v}))$, то есть

$$\forall i \in I_{\boldsymbol{v}} : \langle \boldsymbol{a}_i, \boldsymbol{c} \rangle = 0,$$

то мы получим $\boldsymbol{v} = \boldsymbol{w}$. В этом случае на шаге 16 генерируется новый случайный вектор $\boldsymbol{c}$ и повторно вычисляются точки $\boldsymbol{u}$ и $\boldsymbol{w}$ (шаги 17, 18). Если на шаге 19 $\boldsymbol{v} \neq \boldsymbol{w}$, то происходит выход из цикла **repeat/until**. При $I_{\boldsymbol{v}} = \emptyset$ вместо шагов 15–19 выполняется шаг 22, который

Алгоритм 2 Прохождение грани

Require: $\boldsymbol{v} \in M$; $\dim(\text{face}(\boldsymbol{v})) > 0$

1: **function** MoveAlongFace($\boldsymbol{v}$)
2: $I_{\boldsymbol{v}} := \bar{I}$
3: **for** $i = 1, \ldots, m$ **do**
4: **if** $\boldsymbol{v} \in \hat{H}_i$ **then**
5: $I_{\boldsymbol{v}} := I_{\boldsymbol{v}} \cup \{i\}$
6: **end if**
7: **end for**
8: $I'_{\boldsymbol{v}} := I_{\boldsymbol{v}}$
9: **for** $i \in I'_{\boldsymbol{v}}$ **do**
10: **if** $\text{rank}(A'_{I_v - \{i\}}) = \text{rank}(A'_{I_v})$ **then**
11: $I_{\boldsymbol{v}} := I_{\boldsymbol{v}} - \{i\}$
12: **end if**
13: **end for**
14: **if** $I_{\boldsymbol{v}} \neq \emptyset$ **then**
15: **repeat**
16: $\boldsymbol{c} := \text{RandomNonZeroVector}()$
17: $\boldsymbol{u} := \boldsymbol{v} + \delta \boldsymbol{c}/\|\boldsymbol{c}\|$
18: $\boldsymbol{w} := \boldsymbol{\pi}_{I_v}(\boldsymbol{u})$
19: **until** $\boldsymbol{v} \neq \boldsymbol{w}$
20: **end if**
21: **if** $I_{\boldsymbol{v}} = \emptyset$ **then**
22: $\boldsymbol{w} := \boldsymbol{\pi}_1(\boldsymbol{v})$
23: **end if**
24: **if** $\langle \boldsymbol{c}, \boldsymbol{w} \rangle > \langle \boldsymbol{c}, \boldsymbol{v} \rangle$ **then**
25: $\boldsymbol{d} := \boldsymbol{w} - \boldsymbol{v}$
26: **else**
27: $\boldsymbol{d} := \boldsymbol{v} - \boldsymbol{w}$
28: **end if**
29: $\boldsymbol{v}_{next} := \text{Jump}(\boldsymbol{v}, \boldsymbol{d})$
30: **return** $\boldsymbol{v}_{next}$
31: **end function**

в качестве $\boldsymbol{w}$ выбирает ортогональную проекцию точки $\boldsymbol{v}$ на граничную гиперплоскость с индексом 1 по следующей формуле [12]:

$$\boldsymbol{\pi}_1(\boldsymbol{v}) = \boldsymbol{v} - \frac{\langle \boldsymbol{a}_1, \boldsymbol{v} \rangle - b_1}{\|\boldsymbol{a}_1\|^2} \boldsymbol{a}_1.$$

Из $I_{\boldsymbol{v}} = \emptyset$ следует, что точка $\boldsymbol{v}$ является внутренней точкой допустимого многогранника M. Поэтому $\boldsymbol{v} \neq \boldsymbol{w}$. Шаги 24–28 вычисляют направляющий вектор $\boldsymbol{d} \neq \boldsymbol{0}$, ведущий к увеличению значения целевой функции. В завершение на шаге 29 с помощью вызова функции $\text{Jump}(\boldsymbol{v}, \boldsymbol{d})$ осуществляется переход по этому направлению к новой граничной точке $\boldsymbol{v}_{next}$, которая возвращается в качестве результата на шаге 30.

Функция MoveAlongFace$(\cdot)$ в качестве аргумента требует точку $\boldsymbol{v}$, принадлежащую допустимому многограннику M. Такую точку можно получить с помощью метода релаксации Агмона—Моцкина—Шёнберга [17, 18]. При этом систему общего вида (6) необходимо будет привести к стандартному виду (1).

Реализация алгоритма VeSP, выполняющего поиск вершины допустимого многогранника M, определяемого системой ограничений (6), представлена в виде алгоритма 3.

Алгоритм 3 VeSP (Vertex Seeking by Projecting)

Require: $\boldsymbol{v}_0 \in M$

1: $t := 0$
2: **while** $\dim(\text{face}(\boldsymbol{v}_t)) > 0$ **do**
3: $\quad \boldsymbol{v}_{t+1} := \text{MoveAlongFace}(\boldsymbol{v}_t)$
4: $\quad t := t + 1$
5: **end while**
6: **output** $\boldsymbol{v}_t$
7: **stop**

Прокомментируем шаги этого алгоритма. Шаг 1 устанавливает счетчик итераций t в значение 0. Шаг 2 открывает цикл **while** вычисления вершины многогранника. Если размерность грани, на которой лежит точка $\boldsymbol{v}_t$, больше нуля, выполняется шаг 3, который с помощью функции MoveAlongFace$(\cdot)$ (см. алгоритм 2) осуществляет переход к точке $\boldsymbol{v}_{t+1}$, лежащей на грани меньшей размерности. После этого выполняется шаг 4, увеличивающий счетчик итераций t на единицу, и выполняется переход на начало цикла **while**. Если условие цикла **while** на шаге 2 не выполняется, то точка $\boldsymbol{v}_t$ является вершиной допустимого многогранника M. В этом случае выполняется переход на шаг 6, который выводит точку $\boldsymbol{v}_t$ в качестве результата. Шаг 7 завершает работу алгоритма.

Сходимость алгоритма 3 к вершине допустимого многогранника за конечное число итераций гарантируется следующей теоремой.

Теорема 1. (Сходимость алгоритма VeSP) Пусть система ограничений (6) является регулярной, то есть выполняются условия определения 1. Обозначим через

$$\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_t, \ldots \tag{42}$$

последовательность точек, генерируемых алгоритмом 3. Тогда эта последовательность сходится к точке, являющейся вершиной допустимого многогранника M, не более, чем за n итераций, где n — размерность пространства:

$$\exists t' \leqslant n : \dim(\text{face}(\boldsymbol{v}_{t'})) = 0.$$

Доказательство. Если $\dim(\text{face}(\boldsymbol{v}_0)) = 0$ то точка $\boldsymbol{v}_0$ является вершиной допустимого многогранника M. В этом случае алгоритм 3 выводит $\boldsymbol{v}_0$ в качестве результата и на этом заканчивает свою работу.

Пусть начальная точка $\boldsymbol{v}_0$ вершиной не является. В этом случае последовательность (42) содержит более одного элемента. Пусть $\boldsymbol{v}_t$ — произвольный не конечный элемент по-

следовательности (42), то есть $\boldsymbol{v}_t$ не является вершиной:

$$\dim(\text{face}(\boldsymbol{v}_t)) > 0. \tag{43}$$

В соответствии с (35) имеем

$$\dim(\text{face}(\boldsymbol{v}_t)) = n - \min(k + l, n),$$

где l — количество граничных гиперплоскостей, проходящих через точку $\boldsymbol{v}_t$. В силу (43) отсюда получаем

$$k + l < n \tag{44}$$

и

$$\dim(\text{face}(\boldsymbol{v}_t)) = n - k - l. \tag{45}$$

Шаг 3 алгоритма 3 с помощью вызова функции MoveAlongFace($\boldsymbol{v}_t$), реализуемой алгоритмом 2, генерирует следующий элемент $\boldsymbol{v}_{t+1}$. Проанализируем работу алгоритма 2. Точке $\boldsymbol{v}_t$ в этом алгоритме соответствует точка $\boldsymbol{v}$:

$$\boldsymbol{v} \equiv \boldsymbol{v}_t. \tag{46}$$

Из (45) и (46) следует

$$\dim(\text{face}(\boldsymbol{v})) = n - k - l. \tag{47}$$

На шагах 2–7 алгоритма 2 формируется множество $I_{\boldsymbol{v}}$, включающее в себя индексы всех опорных и граничных гиперплоскостей, проходящих через точку $\boldsymbol{v}$. Так как по условию теоремы система ограничений является регулярной, то из (44) и (24) следует, что множество $I_{\boldsymbol{v}}$ не содержит индексов избыточных уравнений.

Предположим сначала, что $I_{\boldsymbol{v}} \neq \emptyset$. В этом случае выполняются шаги 15–19. Обозначим через $S_{\boldsymbol{v}}$ подпространство, являющееся аффинной оболочкой грани face($\boldsymbol{v}$):

$$S_{\boldsymbol{v}} = \bigcap_{i \in I_{\boldsymbol{v}}} H_i.$$

На шагах 15–19 вычисляется точка $\boldsymbol{w} \in S_{\boldsymbol{v}}$, отличная от точки $\boldsymbol{v}$. С использованием этих точек шаги 24–28 формируют ненулевой направляющий вектор $\boldsymbol{d} = \boldsymbol{w} - \boldsymbol{v}$, задающий луч

$$\text{ray}(\boldsymbol{v}, \boldsymbol{d}) = \{\, \boldsymbol{x} \in \mathbb{R}^n \,|\, \boldsymbol{x} = \boldsymbol{v} + \lambda \boldsymbol{d},\ \lambda \geqslant 0 \}\,.$$

По построению этот луч принадлежит подпространству $S_{\boldsymbol{v}}$:

$$\text{ray}(\boldsymbol{v}, \boldsymbol{d}) \subseteq S_{\boldsymbol{v}}. \tag{48}$$

На шаге 29 с помощью вызова функции Jump($\boldsymbol{v}, \boldsymbol{d}$) (см. алгоритм 1) вычисляется точка $\boldsymbol{v}_{next} \in M$, являющаяся пересечением луча ray($\boldsymbol{v}, \boldsymbol{d}$) с ближайшей граничной гиперплоскостью $\hat{H}_{i'}$ такой, что $i' \notin I_{\boldsymbol{v}}$. Такая гиперплоскость существует в силу ограниченности допустимого многогранника M. Из (48) следует, что через точку $\boldsymbol{v}_{next}$ проходит не менее $l' = l + 1$ граничных гиперплоскостей. С учетом (35) и (44) отсюда получаем

$$\dim(\text{face}(\boldsymbol{v}_{next})) \leqslant n - k - l - 1. \tag{49}$$

Сопоставляя (47) и (49), имеем

$$\dim(\text{face}(\boldsymbol{v})) > \dim(\text{face}(\boldsymbol{v}_{next})).$$

Теперь предположим, что $I_{\boldsymbol{v}} = \emptyset$, то есть $k = 0$ и $l = 0$. В соответствии со следствием 4 отсюда следует

$$\dim(\text{face}(\boldsymbol{v})) = n. \tag{50}$$

В этом случае выполняется шаг 22, в результате которого мы получаем точку $\boldsymbol{w}$ на граничной гиперплоскости $\hat{H}_1$. Так как через $\boldsymbol{v}$ не проходит ни одной граничной гиперплоскости, то $\boldsymbol{v} \neq \boldsymbol{w}$. Шаги 24–28 формируется ненулевой направляющий вектор $\boldsymbol{d} = \boldsymbol{w} - \boldsymbol{v}$, задающий луч $\text{ray}(\boldsymbol{v}, \boldsymbol{d})$. На шаге 29 с помощью вызова функции $\text{Jump}(\boldsymbol{v}, \boldsymbol{d})$ (см. алгоритм 1) вычисляется точка $\boldsymbol{v}_{next} \in M$, являющаяся пересечением луча $\text{ray}(\boldsymbol{v}, \boldsymbol{d})$ с ближайшей граничной гиперплоскостью. Таким образом, через точку $\boldsymbol{v}_{next}$ проходит не менее одной граничной гиперплоскости. В соответствии с (35) отсюда следует

$$\dim(\text{face}(\boldsymbol{v}_{next})) \leqslant n - 1. \tag{51}$$

Сопоставляя (50) и (51), и в этом случае имеем

$$\dim(\text{face}(\boldsymbol{v})) > \dim(\text{face}(\boldsymbol{v}_{next})). \tag{52}$$

Точка $\boldsymbol{v}_{next}$ на шаге 30 возвращается как результат вызова функции $\text{MoveAlongFace}(\boldsymbol{v}_t)$ в алгоритме 3 на шаге 3. Поэтому

$$\boldsymbol{v}_{next} \equiv \boldsymbol{v}_{t+1}. \tag{53}$$

Из 46, 52 и 53 следует

$$\dim(\text{face}(\boldsymbol{v}_t)) > \dim(\text{face}(\boldsymbol{v}_{t+1})). \tag{54}$$

Таким образом, последовательности (42) соответствует убывающая последовательность неотрицательных целых чисел

$$\dim(\text{face}(\boldsymbol{v}_0)) > \dim(\text{face}(\boldsymbol{v}_1)) > \cdots > \dim(\text{face}(\boldsymbol{v}_t)) > \ldots,$$

ограниченная сверху числом, равным размерности пространства n. Следовательно, последовательность (42) имеет длину не более $n + 1$. Осталось заметить, что в соответствии с условием выхода из цикла **while** (шаг 2), последним элементом последовательности (42) будет некоторая вершина допустимого многогранника M. Теорема доказана. □

3. Реализация и вычислительные эксперименты

Нами была выполнена реализация алгоритма VeSP на языке C++, исходные коды которой свободно доступны на `https://github.com/leonid-sokolinsky/VeSP`. Мы протестировали алгоритм VeSP на эталонных задачах из репозитория Netlib-LP [19, 20], доступного по адресу `https://netlib.org/lp/data`. Тестирование выполнялось на персональном компьютере с процессором Intel Core i7-13620H 2.40 GHz, DDR5 32 ГБ. Результаты тестирования представлены в табл. 1.

В заголовке таблицы использованы следующие обозначения: в колонке «Задача» указано имя задачи ЛП из репозитория Netlib-LP; m — количество ограничений; n — количество переменных (размерность пространства решений); $\dim(M)$ — размерность многогранника

Таблица 1. Результаты тестирования алгоритма VeSP на задачах Netlib-LP

Задача	m	n	$\dim(M)$	$\langle \boldsymbol{c}, \boldsymbol{u}_0 \rangle$	Итер.	$\langle \boldsymbol{c}, \boldsymbol{v} \rangle$	Время	$\operatorname{dist}(\boldsymbol{v}, M)$
adlittle	56	97	82	0.106230×10^{8}	44	0.716431×10^{6}	0	2.3×10^{-7}
afiro	27	32	24	0.532704×10^{2}	6	-0.309501×10^{2}	0	9.9×10^{-11}
beaconfd	173	262	122	0.338431×10^{5}	35	0.337284×10^{5}	2	1.5×10^{-11}
blend*	74	83	40	-0.508862×10^{1}	18	-0.158981×10^{2}	775	1.1×10^{-12}
fit1d	24	1026	1025	-0.483981×10^{4}	530	-0.899543×10^{4}	614	4.0×10^{-7}
grow7	140	301	161	-0.207633×10^{8}	119	-0.439694×10^{8}	2	9.7×10^{-6}
israel	174	142	142	0.307907×10^{7}	45	-0.102957×10^{6}	0	5.9×10^{-7}
kb2	43	41	25	-0.560444	2	-0.174527×10^{3}	0	2.2×10^{-10}
recipe	91	180	92	-0.230604×10^{3}	9	-0.262820×10^{3}	0	1.5×10^{-12}
sc105	104	103	58	-0.537203×10^{-2}	7	-0.981874×10^{1}	0	3.6×10^{-11}
sc50a	49	48	28	-0.710608×10^{-1}	7	-0.414361×10^{2}	0	9.0×10^{-12}
sc50b	48	48	28	-0.107994	13	-0.478109×10^{2}	0	4.9×10^{-12}
scagr7	129	140	56	-0.174392×10^{7}	22	-0.202156×10^{7}	0	3.9×10^{-8}
share2b	96	79	66	-0.368733×10^{3}	10	-0.389506×10^{3}	0	1.3×10^{-7}
stocfor1	117	111	48	-0.254283×10^{5}	8	-0.254636×10^{5}	0	1.1×10^{-12}

*Применен итерационный алгоритм Качмажа.

допустимых решений, вычисляемая по формуле $\dim(M) = n - \operatorname{rank}(\bar{A})$, где $\bar{A}$ — матрица коэффициентов уравнений, входящих в систему ограничений; $\langle \boldsymbol{c}, \boldsymbol{u}_0 \rangle$ — значение целевой функции в начальной точке $\boldsymbol{u}_0 \in M$; в колонке «Итер.» указано количество итераций, выполненных алгоритмом 3 при поиске вершины многогранника M; $\langle \boldsymbol{c}, \boldsymbol{v} \rangle$ — значение целевой функции в найденной вершине $\boldsymbol{v}$; «Время» — затраченное время в секундах; $\operatorname{dist}(\boldsymbol{v}, M)$ — точность решения, вычисляемая как евклидово расстояние от точки $\boldsymbol{v}$ до многогранника M.

Начальная точка $\boldsymbol{u}_0 \in M$ вычислялась с помощью написанных нами программ FIP и Quest. Программа FIP (Feasible point Iterative Projection) стартует из точки $\boldsymbol{x}_0 = \boldsymbol{0}$ и находит точку $\boldsymbol{z}_0$, принадлежащую допустимому многограннику M. В основе программы FIP лежит метод релаксаций Агмона—Моцкина—Шёнберга [17, 18]. Исходные коды этой программы доступны на `https://github.com/leonid-sokolinsky/FIP`. Полученная точка $\boldsymbol{z}_0 \in M$ использовалась программой Quest для нахождения так называемой точки апекса $\boldsymbol{u}_0$, которая представляла собой «грубое» приближение к решению задачи ЛП [21]. Эта точка использовалась алгоритмом VeSP в качестве начального приближения. Исходные коды программы Quest доступны на `https://github.com/leonid-sokolinsky/Quest`.

Проведенные эксперименты показали, что алгоритм VeSP успешно нашел вершину допустимого многогранника для всех 15 задач, выбранных из репозитория Netlib-LP. Время вычислений для задач с числом переменных $n < 200$, за исключением задачи «blend», составило менее половины секунды. Задачи «beaconfd» и «grow7» с количеством переменных 262 и 301 соответственно потребовали 2 секунды процессорного времени. При решении задачи «fit1d», содержащей 1026 переменных, алгоритм VeSP в процессе поиска вершины выполнил 513 итераций, что потребовало более 10 минут. Таким образом, можно сделать вывод, что время работы алгоритма VeSP существенно зависит от размерности задачи. Что касается задачи «blend» с числом переменных 83, для нее не удалось вычислить обратную матрицу в формуле (37) по причине потери точности при использовании 64-разрядной вещественной арифметики. Поэтому для приближенного вычисления ортогональных проекций (шаги 18, 22 алгоритма 2) пришлось использовать итерационный алгоритм Качмажа. По этой причине 18 итераций, выполненных алгоритмом VeSP для этой задачи, заняли 13 минут.

При этом более 99% процессорного времени было затрачено на вычисление ортогональных проекций. Это объясняется тем, что итерационный алгоритм Качмажа имеет линейную скорость сходимости [22]:

$$\|\boldsymbol{x}_k - \boldsymbol{\pi}_J(\boldsymbol{x}_0)\| \leqslant \theta \, \|\boldsymbol{x}_{k-1} - \boldsymbol{\pi}_J(\boldsymbol{x}_0)\|$$

с вещественной константой $0 < \theta < 1$. Здесь $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{k-1}, \boldsymbol{x}_k, \ldots\}$ — последовательные приближения к ортогональной проекции $\boldsymbol{\pi}_J(\boldsymbol{x}_0)$ точки $\boldsymbol{x}_0$ на подпространство S_J, вычисляемые алгоритмом Качмажа. Константа θ зависит только от углов между гиперплоскостями H_i $(i \in J)$. При малых углах значение θ приближается к единице, и скорость сходимости стремится к нулю. Проведенные эксперименты также показали, что точность вычисления координат вершины допустимого многогранника находится в обратной пропорции относительно количества выполненных итераций.

Заключение

В статье предложен новый проекционный алгоритм VeSP для нахождения вершины многогранника допустимых решений системы линейных ограничений. Под системой линейных ограничений понимается система общего вида, включающая в себя как линейные неравенства, так и линейные уравнения. Подобные системы ограничений характерны для задач линейного программирования. В качестве стартовой точки алгоритм VeSP использует произвольную точку многогранника допустимых решений. Алгоритм VeSP применим к любым системам линейных ограничений, имеющим непустую ограниченную область допустимых решений. Доказано утверждение, позволяющее для заданной допустимой точки вычислить ее собственную грань (грань наименьшей размерности, содержащую эту точку). В качестве следствий получены достаточные условия для того, чтобы собственная грань допустимой точки была вершиной, ребром или многогранником допустимых решений. Представлено формализованное описание алгоритма VeSP на псевдокоде. Также представлены формализованные описания двух подпрограмм-функций, используемых алгоритмом VeSP: подпрограмма-функция, осуществляющая перемещение по направляющему вектору от заданной допустимой точки к максимально удаленной допустимой точке, и подпрограмма-функция прохождения грани многогранника. Доказана теорема сходимости алгоритма VeSP к вершине многогранника допустимых решений за количество итераций, не превышающих размерность пространства. Выполнена реализация алгоритма VeSP на языке программирования C++. Указанная реализация была протестирована на эталонных задачах из репозитория Netlib-LP. Эксперименты показали, что алгоритм VeSP способен эффективно находить базисное решение для реальных задач ЛП. Основным преимуществом алгоритма VeSP является то, что он гарантированно находит вершину многогранника допустимых решений не более, чем за n итераций. Алгоритмам, основанным на симплекс-методе и методе исключения переменных Фурье—Моцкина, для этого может понадобиться 2^n и $2^{n/2}$ итераций соответственно.

Обозначения

n	число переменных в системе ограничений (размерность пространства)
m	количество неравенств в системе ограничений
k	количество уравнений в системе ограничений
$\mathbb{R}^d$	вещественное евклидово пространство размерности d

$\langle \cdot, \cdot \rangle$	скалярное произведение двух векторов
$\hat{A}$	матрица коэффициентов неравенств: $\hat{A} \in \mathbb{R}^{m \times n}$
$\bar{A}$	матрица коэффициентов уравнений: $\bar{A} \in \mathbb{R}^{k \times n}$
$\hat{\boldsymbol{b}}$	столбец правых частей неравенств: $\hat{\boldsymbol{b}} \in \mathbb{R}^m$
$\bar{\boldsymbol{b}}$	столбец правых частей уравнений: $\bar{\boldsymbol{b}} \in \mathbb{R}^k$
$\boldsymbol{a}_i$	i-тая строка матрицы $\begin{bmatrix} \hat{A} \\ \bar{A} \end{bmatrix}$ $(i = 1, \ldots, m + k)$
$\hat{I}$	индексы (номера) строк матрицы $\hat{A}$: $\hat{I} = \{1, \ldots, m\}$
$\bar{I}$	индексы (номера) строк матрицы $\bar{A}$: $\bar{I} = \{m + 1, \ldots, m + k\}$
P_i	полупространство, определяемое формулой $\langle \boldsymbol{a}_i, \boldsymbol{x} \rangle \leqslant b_i$ при $i \in \hat{I}$
$\hat{H}_i$	граничная гиперплоскость полупространства P_i
$\bar{H}_i$	опорная гиперплоскость, определяемая уравнением $\langle \boldsymbol{a}_i, \boldsymbol{x} \rangle = b_i$
$\hat{M}$	граничный многогранник (пересечение всех полупространств P_i)
$\bar{S}$	опорное подпространство (пересечение всех опорных гиперплоскостей $\bar{H}_i$)
M	допустимый многогранник (область допустимых решений): $M = \hat{M} \cap \bar{S}$
$\mathrm{aff}(X)$	аффинная оболочка множества X
$\dim(X)$	размерность множества X: $\dim(X) = \dim(\mathrm{aff}(X))$
$\mathrm{face}(\boldsymbol{v})$	собственная грань точки $\boldsymbol{v} \in M$
$\mathrm{rank}(A)$	ранг матрицы A
$\|\cdot\|$	евклидова норма

Литература

1. Dantzig G.B. Linear programming and extensions. Princeton, N.J.: Princeton university press, 1998. 656 p.
2. Жулев А., Соколинский Л. AlEM: новый параллельный алгоритм линейного программирования для кластерных вычислительных систем // PREPRINTS.RU. 2025. С. 1–19. DOI: 10.24108/preprints-3113529.
3. Avis D. A Revised Implementation of the Reverse Search Vertex Enumeration Algorithm // Polytopes - Combinatorics and Computation. DMV Seminar, vol 29 / ed. by G. Kalai, G. Ziegler. Basel: Birkhauser, 2000. P. 177–198. DOI: 10.1007/978-3-0348-8438-9_9.
4. Assad C.L., Morales G., Arica J. Vertex Enumeration of Polyhedra // Pesquisa Operacional. 2022. Vol. 42: e25457. P. 1–20. DOI: 10.1590/0101-7438.2022.042.00254570.
5. Схрейвер А. Теория линейного и целочисленного программирования: в 2 т. Т. 1. Москва: Мир, 1991. 360 с.
6. Klee V., Minty G.J. 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, NY, USA: Academic Press, 1972. P. 159–175.
7. Khachiyan L. Fourier–Motzkin Elimination Method // Encyclopedia of Optimization / ed. by C. Floudas, P. Pardalos. Boston, MA: Springer, 2008. P. 1074–1077. DOI: 10.1007/978-0-387-74759-0_187.
8. Duffin R.J. On fourier's analysis of linear inequality systems // Pivoting and Extensions / ed. by M. Balinski. Berlin, Heidelberg: Springer, 1974. P. 71–95. DOI: 10.1007/BFB0121242.

9. Motzkin T.S. Contributions to the Theory of Linear Inequalities // Theodore S. Motzkin: Selected Papers / ed. by D. Cantor, B. Gordon, B.L. Rothschild. Boston: Birkhauser, 1983. P. 1–80.

10. Gritzmann P., Klee V. Mathematical Programming and Convex Geometry // Handbook of Convex Geometry, Vol. A / ed. by P.M. Gruber, J.M. Wills. Amsterdam: Elsevier, 1993. P. 627–674.

11. Циглер Г.М. Теория многогранников. М.: МЦНМО, 2014. 568 с.

12. Murty K.G. Computational and Algorithmic Linear Algebra and n-Dimensional Geometry. World Scientific, 2011. xxi, 552 p. DOI: 10.1142/8261.

13. 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.

14. Chen X. The Kaczmarz algorithm, row action methods, and statistical learning algorithms // Frames and Harmonic Analysis. Contemporary Mathematics, vol. 706 / ed. by Y. Kim, S. Narayan, G. Picioroaga, E. Weber. Providence, Rhode Island: American Mathematical Society, 2018. P. 115–128. DOI: 10.1090/CONM/706.

15. Ольховский Н.А., Соколинский Л.Б. О новом методе линейного программирования // Вычислительные методы и программирование. 2023. Т. 24, № 4. С. 408–429. DOI: 10.26089/NumMet.v24r428.

16. 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.

17. 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.

18. 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.

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

20. 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.

21. 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.

22. Deutsch F. Rate of Convergence of the Method of Alternating Projections // Parametric Optimization and Approximation / ed. by B. Brosowski, F. Deutsch. Basel: Birkhauser Verlag, 1985. P. 96–107. DOI: 10.1007/978-3-0348-6253-0_7.

Жулев Александр Эдуардович, аспирант кафедры системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

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 прикладной математики и программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse250301

ON CALCULATING A VERTEX OF FEASIBLE SOLUTIONS POLYTOPE OF LINEAR CONSTRAINT SYSTEM

© 2025 A.E. Zhulev, L.B. Sokolinsky, I.M. Sokolinskaya

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

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

Received: 14.06.2025

The article is devoted to a new algorithm for calculating a vertex of polytope being the feasible region of linear constraint system. The algorithm called VeSP starts at an arbitrary point of the polytope and, moving along its faces, stops at some vertex. To calculate the movement direction along the face, it uses the projection method. The idea of this method is as follows. For the current approximation point, an affine subspace is calculated, which is the affine hull of the face containing the point. A non-zero vector is added to the current approximation point. This gives an external point relative to the current affine subspace. The orthogonal projection of the external point onto the current affine subspace is calculated using a known analytical formula. The projection point determines the direction of movement along the edge to its boundary, which gives the next approximation point. Each movement reduces the dimension of the current face. Thus, we arrive at a zero-dimensional face, which is the vertex of the polytope. A formal description of the VeSP algorithm is provided. The convergence of the VeSP algorithm to a polytope vertex in a finite number of iterations is proved. This number does not exceed the space dimension. An information about the implementation of the VeSP algorithm in C++ is provided. The results of computational experiments with real problems from the Netlib-LP collection are described. For all test problems, the VeSP algorithm successfully found the vertex of the polytope in a finite number of iterations that did not exceed the space dimension. For most problems, finding the vertex took less than one second on a commodity personal computer.

Keywords: linear constraints, feasible solutions polytope, vertex calculation, projection method, VeSP algorithm.

FOR CITATION

Zhulev A.E., Sokolinsky L.B., Sokolinskaya I.M. On Calculating a Vertex of Feasible Solutions Polytope of Linear Constraints System. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2025. Vol. 14, no. 3. P. 5–27. (in Russian) DOI: 10.14529/cmse250301.

This paper is distributed under the terms of the Creative Commons Attribution-Non Commercial 4.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

References

1. Dantzig G.B. Linear programming and extensions. Princeton, N.J.: Princeton university press, 1998. 656 p.
2. Zhulev A., Sokolinsky L. AlEM: a new parallel algorithm for linear programming on cluster computing systems. PREPRINTS.RU. 2025. P. 1–19. (in Russian) DOI: 10.24108/preprints-3113529.

3. Avis D. A Revised Implementation of the Reverse Search Vertex Enumeration Algorithm. Polytopes - Combinatorics and Computation. DMV Seminar, vol 29 / ed. by G. Kalai, G. Ziegler. Basel: Birkhauser, 2000. P. 177–198. DOI: 10.1007/978-3-0348-8438-9_9.
4. Assad C.L., Morales G., Arica J. Vertex Enumeration of Polyhedra. Pesquisa Operacional. 2022. Vol. 42: e25457. P. 1–20. DOI: 10.1590/0101-7438.2022.042.00254570.
5. Schrijver A. Theory of Linear and Integer Programming. Chichester, New York, Brisbane, Tofonfo, Singapore: Wiley, Sons, 1998. 484 p.
6. Klee V., Minty G.J. 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, NY, USA: Academic Press, 1972. P. 159–175.
7. Khachiyan L. Fourier–Motzkin Elimination Method. Encyclopedia of Optimization / ed. by C. Floudas, P. Pardalos. Boston, MA: Springer, 2008. P. 1074–1077. DOI: 10.1007/978-0-387-74759-0_187.
8. Duffin R.J. On fourier's analysis of linear inequality systems. Pivoting and Extensions / ed. by M. Balinski. Berlin, Heidelberg: Springer, 1974. P. 71–95. DOI: 10.1007/BFB0121242.
9. Motzkin T.S. Contributions to the Theory of Linear Inequalities. Theodore S. Motzkin: Selected Papers / ed. by D. Cantor, B. Gordon, B.L. Rothschild. Boston: Birkhauser, 1983. P. 1–80.
10. Gritzmann P., Klee V. Mathematical Programming and Convex Geometry. Handbook of Convex Geometry, Vol. A / ed. by P.M. Gruber, J.M. Wills. Amsterdam: Elsevier, 1993. P. 627–674.
11. Ziegler G.M. Lectures on Polytopes. Vol. 152. New York, NY: Springer New York, 1995. XI, 370 p. Graduate Texts in Mathematics. DOI: 10.1007/978-1-4613-8431-1.
12. Murty K.G. Computational and Algorithmic Linear Algebra and n-Dimensional Geometry. World Scientific, 2011. xxi, 552 p. DOI: 10.1142/8261.
13. 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.
14. Chen X. The Kaczmarz algorithm, row action methods, and statistical learning algorithms. Frames and Harmonic Analysis. Contemporary Mathematics, vol. 706 / ed. by Y. Kim, S. Narayan, G. Picioroaga, E. Weber. Providence, Rhode Island: American Mathematical Society, 2018. P. 115–128. DOI: 10.1090/CONM/706.
15. Olkhovsky N.A., Sokolinsky L.B. Surface Movement Method for Linear Programming. Lobachevskii Journal of Mathematics. 2024. Vol. 45, no. 10. P. 5061–5079. DOI: 10.1134/S1995080224605745.
16. 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.
17. 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.
18. 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.

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

20. 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.

21. Sokolinsky L.B., Sokolinskaya I.M. Apex Method: A New Scalable Iterative Method for Linear Programming. Mathematics. 2023. Vol. 11, no. 7. P. 1–28. DOI: 10.3390/MATH11071654.

22. Deutsch F. Rate of Convergence of the Method of Alternating Projections. Parametric Optimization and Approximation / ed. by B. Brosowski, F. Deutsch. Basel: Birkhauser Verlag, 1985. P. 96–107. DOI: 10.1007/978-3-0348-6253-0_7.