УДК 519.852, 004.032.26 **DOI: 10.14529/cmse220103**

ВИЗУАЛЬНОЕ ПРЕДСТАВЛЕНИЕ МНОГОМЕРНЫХ ЗАДАЧ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

© 2022 Н.А. Ольховский, Л.Б. Соколинский

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

Поступила в редакцию: 10.03.2022

В статье строится n-мерная математическая модель визуального представления задачи линейного программирования. Эта модель позволит использовать аппарат искусственных нейронных сетей для решения многомерных задач линейной оптимизации, допустимая область которых является ограниченным непустым множеством. Для визуализации задачи линейного программирования вводится целевая гиперплоскость, ориентация которой определяется градиентом линейной целевой функции: градиент является нормалью к целевой гиперплоскости. В случае поиска максимума целевая гиперплоскость располагается таким образом, чтобы значение целевой функции во всех ее точках превосходило значение целевой функции во всех точках допустимой области, представляющей собой ограниченный выпуклый многогранник. Для произвольной точки целевой гиперплоскости определяется целевая проекция на многогранник: чем ближе точка целевой проекции к целевой гиперплоскости, тем больше значение целевой функции в этой точке. На основе целевой гиперплоскости строится конечное регулярное множество точек, называемое рецептивным полем. С помощью целевых проекций строится образ многогранника, включающий в себя точки рецептивного поля и расстояния до соответствующих точек поверхности многогранника. На основе предложенной модели строится параллельный алгоритм визуализации задачи линейного программирования. Дается аналитическая оценка его масштабируемости. Приводятся сведения о программной реализации и результаты масштабных вычислительных экспериментов, подтверждающие эффективность предложенных подходов.

Ключевые слова: линейное программирование, n-мерная визуализация, математическая модель, параллельный алгоритм, BSF-каркас.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

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.

Введение

Быстрое развитие технологий накопления и обработки больших данных [1, 2] привело к появлению оптимизационных математических моделей в виде сверхбольших задач линейного программирования (ЛП) [3]. Такие задачи возникают в индустрии, экономике, логистике, статистике, квантовой физике и других областях [4—8]. Классическое программное обеспечение во многих случаях не позволяет решить подобные масштабные задачи ЛП за приемлемое время [9]. Вместе с тем в ближайшие 2–3 года появятся вычислительные системы экзафлопсного уровня производительности [10], потенциально способные решать подобные задачи. В соответствии с этим актуальной является задача разработки новых эффективных методов для решения сверхбольших задач ЛП с помощью экзамасштабных вычислительных систем.

До настоящего времени одним из самых распространенных способов решения задачи ЛП являлся класс алгоритмов, предложенных и разработанных Данцигом на основе симплекс-метода [11]. Симплекс-метод оказался эффективным для решения большого клас-

са задач ЛП. Однако симплекс-метод имеет некоторые фундаментальные особенности, ограничивающие его применение для больших задач ЛП. Во-первых, в определенных случаях симплекс-методу приходится перебирать все вершины симплекса, что соответствует экспоненциальной временной сложности [12]. Во-вторых, симплекс-метод в большинстве случаев удовлетворительно решает задачи ЛП, содержащие до 50000 переменных, однако на больших задачах часто наблюдается потеря точности, которая не может быть компенсирована даже путем применения таких ресурсоемких процедур, как «аффинное масштабирование» или «итерационное уточнение» [13]. В-третьих, в общем случае симплекс метод плохо масштабируется на многопроцессорных системах с распределенной памятью. Были предприняты многочисленные попытки построить масштабируемую реализацию симплекс-метода, однако они не увенчались успехом [14]. Кармаркар предложил метод внутренних точек [15], способный решать сверхбольшие задачи ЛП с миллионами переменных и миллионами уравнений [16]. Более того, алгоритмы, основанные на методе внутренних точек, являются самокорректирующимися, и поэтому обеспечивают высокую точность вычислений. В качестве недостатка метода внутренних точек следует отметить тот факт, что для начала работы алгоритма необходимо иметь точку, удовлетворяющую всем ограничениям задачи ЛП. Нахождение такой внутренней точки может сводиться к решению дополнительной задачи ЛП. В качестве альтернативы можно указать метод псевдопроекций, основанный на использовании фейеровских отображений [17]. Еще одним существенным недостатком метода внутренних точек является его плохая масштабируемость применительно к многопроцессорным системам с распределенной памятью. Было сделано несколько попыток построить параллельную реализацию для частных случаев (см., например, [18, 19]), но эффективную параллельную реализацию для экзамасштабных многопроцессорных систем в общем случае построить не удалось. В соответствии с этим является актуальным направление исследований, связанное с поиском новых масштабируемых методов решения задач ЛП.

Возможной эффективной альтернативой классическим методам ЛП являются методы оптимизации, основанные на нейросетевых технологиях. Искусственные нейронные сети [20, 21] являются одним из самых обещающих и быстро развивающихся направлений современных информационных технологий. Нейронные сети представляют собой универсальный инструмент, способный решать задачи практически во всех областях. Особенно большие успехи достигнуты в распознавании и анализе изображений с помощью сверточных нейронных сетей [22]. Однако, в научной периодике практически отсутствуют работы, посвященные применению сверточных нейронных сетей для решения задач линейной оптимизации [23]. Это связано с тем, что сверточные нейронные сети ориентированы на распознавание и анализ изображений, но в научной литературе отсутствуют работы по визуальному представлению многомерных задач линейного программирования. Таким образом, вопрос разработки новых нейросетевых моделей и методов применительно к линейной оптимизации остается открытым.

В этой статье мы предприняли попытку построить n-мерную математическую модель визуального представления задачи ЛП, которая позволит использовать аппарат искусственных нейронных сетей для решения многомерных задач линейной оптимизации, допустимая область которых является ограниченным непустым множеством. Метод визуализации, основанный на описываемой модели, обладает высокой вычислительной сложностью. Поэтому мы предлагаем его реализацию в виде параллельного алгоритма, ориентированного на кластерные вычислительные системы. Статья организована следующим образом. Раз-

дел 1 посвящен построению математической модели визуального представления многомерной задачи ЛП. В разделе 2 описывается реализация предложенного метода визуализации в виде параллельного алгоритма и дается аналитическая оценка его масштабируемости. В разделе 3 приводится информация о программной реализации описанного параллельного алгоритма и обсуждаются результаты масштабных вычислительных экспериментов на кластерной вычислительной системе. В заключении суммируются полученные результаты и приводятся направления дальнейших исследований.

1. Математическая модель визуального представления задачи ЛП

Рассмотрим задачу ЛП в следующей форме:

$$\bar{x} = \arg\max\left\{\langle c, x\rangle | Ax \leqslant b, x \in \mathbb{R}^n\right\}, \tag{1}$$

где $c, b \in \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$ и $c \neq \mathbf{0}$. Здесь $\langle\cdot,\cdot\rangle$ обозначает скалярное произведение двух векторов. Мы предполагаем, что ограничение $x \geqslant \mathbf{0}$ также включено в систему $Ax \leqslant b$ в виде неравенств

$$\begin{array}{ccccccccccc}
-x_1 & + & 0 & + & \cdots & \cdots & \cdots & + & 0 & \leqslant & 0;\\
0 & - & x_2 & + & 0 & + & \cdots & + & 0 & \leqslant & 0;\\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots\\
0 & + & \cdots & \cdots & \cdots & + & 0 & - & x_n & \leqslant & 0.
\end{array}$$

Вектор c является градиентом линейной целевой функции

$$f(x) = c_1x_1 + \ldots + c_nx_n. \tag{2}$$

Обозначим через M допустимую область задачи (1):

$$M = \left\{x \in \mathbb{R}^n | Ax \leqslant b\right\}. \tag{3}$$

Везде далее мы предполагаем, что M является непустым ограниченным множеством. Это означает, что M представляет собой выпуклый замкнутый многогранник в пространстве $\mathbb{R}^n$, и множество решений задачи (1) не является пустым.

Пусть $\tilde{a}_i \in \mathbb{R}^n$ — вектор, образованный элементами i-той строки матрицы A. Тогда матричное неравенство $Ax \leqslant b$ можно представить в виде системы неравенств

$$\langle\tilde{a}_i, x\rangle \leqslant b_i, i = 1, \ldots, m. \tag{4}$$

Везде далее мы будем предполагать, что для всех $i = 1, \ldots, m$

$$\tilde{a}_i \neq \mathbf{0}. \tag{5}$$

Обозначим через H_i гиперплоскость, задаваемую уравнением

$$\langle\tilde{a}_i, x\rangle = b_i \ (1 \leqslant i \leqslant m). \tag{6}$$

Таким образом,

$$H_i = \left\{x \in \mathbb{R}^n | \langle\tilde{a}_i, x\rangle = b_i\right\}. \tag{7}$$

Определение 1. Под полупространством H_i^+, *порождаемым гиперплоскостью* H_i, будем понимать полупространство, определяемое формулой

$$H_i^+ = \{x \in \mathbb{R}^n | \langle \tilde{a}_i, x \rangle \leqslant b_i\}. \tag{8}$$

Везде далее мы будем предполагать, что задача (1) является невырожденной, то есть

$$\forall i \neq j : H_i \neq H_j \ (i, j \in \{1, \ldots, m\}). \tag{9}$$

Определение 2. Полупространство H_i^+, порождаемое гиперплоскостью H_i, является *рецессивным* относительно вектора c, если

$$\forall x \in H_i, \forall \lambda \in \mathbb{R}_{>0} : x - \lambda c \in H_i^+ \wedge x - \lambda c \notin H_i. \tag{10}$$

Другими словами, луч, исходящий из гиперплоскости H_i в направлении противоположном вектору c, полностью лежит в H_i^+, но не в H_i.

Утверждение 1. Необходимым и достаточным условием рецессивности полупространства H_i^+ относительно вектора c является условие

$$\langle \tilde{a}_i, c \rangle > 0. \tag{11}$$

Доказательство. Докажем сначала необходимость. Пусть выполняется условие (10). Из (7) следует

$$x = \frac{b_i \tilde{a}_i}{\|\tilde{a}_i\|^2} \in H_i. \tag{12}$$

Из (5) следует

$$\lambda = \frac{1}{\|\tilde{a}_i\|^2} \in \mathbb{R}_{>0}. \tag{13}$$

Сопоставляя (10),(12) и (13) получаем

$$\frac{b_i \tilde{a}_i}{\|\tilde{a}_i\|^2} - \frac{1}{\|\tilde{a}_i\|^2} c \in H_i^+;$$
$$\frac{b_i \tilde{a}_i}{\|\tilde{a}_i\|^2} - \frac{1}{\|\tilde{a}_i\|^2} c \notin H_i.$$

С учетом (7), (8) отсюда следует, что

$$\left\langle \tilde{a}_i, \tfrac{b_i \tilde{a}_i}{\|\tilde{a}_i\|^2} - \tfrac{1}{\|\tilde{a}_i\|^2} c \right\rangle < b_i. \tag{14}$$

Выполнив несложные преобразования неравенства (14), получаем (11).

Доказательство достаточности проведем от противного. Предположим, что имеет место (11) и существуют $x \in H_i$ и $\lambda > 0$ такие, что

$$x - \lambda c \notin H_i^+ \vee x - \lambda c \in H_i.$$

В соответствии с (7), (8) это означает

$$\langle \tilde{a}_i, x - \lambda c \rangle \geqslant b_i,$$

что равносильно

$$\langle \tilde{a}_i, x \rangle - \lambda \langle \tilde{a}_i, c \rangle \geqslant b_i.$$

Так как $\lambda > 0$, то в силу (11) отсюда получаем

$$\langle \tilde{a}_i, x \rangle > b_i,$$

что противоречит предположению $x \in H_i$. □

Определение 3. Зафиксируем точку $z \in \mathbb{R}^n$ такую, что полупространство

$$H_c^+ = \{x \in \mathbb{R}^n | \langle c, x - z \rangle \leqslant 0\} \tag{15}$$

включает в себя многогранник M:

$$M \subset H_c^+.$$

Полупространство H_c^+ в этом случае будем называть *целевым полупространством*, а гиперплоскость H_c, определяемую уравнением

$$H_c = \{x \in \mathbb{R}^n | \langle c, x - z \rangle = 0\}, \tag{16}$$

будем называть *целевой гиперплоскостью*.

Обозначим через $\pi_c(x)$ ортогональную проекцию точки x на целевую гиперплоскость H_c:

$$\pi_c(x) = x - \frac{\langle c, x - z \rangle}{\|c\|^2} c. \tag{17}$$

Здесь $\| \cdot \|$ обозначает евклидову норму. Определим *расстояние* $\rho_c(x)$ от точки $x \in H_c^+$ до целевой гиперплоскости H_c следующим образом:

$$\rho_c(x) = \|\pi_c(x) - x\|. \tag{18}$$

Сопоставляя (15), (17) и (18) находим, что в этом случае расстояние $\rho_c(x)$ может быть вычислено следующим образом:

$$\rho_c(x) = \frac{\langle c, z - x \rangle}{\|c\|}. \tag{19}$$

Справедливо следующее утверждение.

Утверждение 2. Для любых $x, y \in H_c^+$

$$\rho_c(x) \leqslant \rho_c(y) \Leftrightarrow \langle c, x \rangle \geqslant \langle c, y \rangle.$$

Доказательство. Используя (19), получаем

$$\begin{aligned}
&\rho_c(x) \leqslant \rho_c(y) \Leftrightarrow \frac{\langle c, z-x\rangle}{\|c\|} \leqslant \frac{\langle c, z-y\rangle}{\|c\|} \\
&\Leftrightarrow \langle c, z-x\rangle \leqslant \langle c, z-y\rangle \\
&\Leftrightarrow \langle c, z\rangle + \langle c, -x\rangle \leqslant \langle c, z\rangle + \langle c, -y\rangle \\
&\Leftrightarrow \langle c, -x\rangle \leqslant \langle c, -y\rangle \\
&\Leftrightarrow \langle c, x\rangle \geqslant \langle c, y\rangle .
\end{aligned}$$

□

В соответствии с утверждением 2 задача (1) эквивалентна следующей задаче:

$$\bar{x} = \arg\min \{\rho_c(x) | x \in M\} . \tag{20}$$

Определение 4. Пусть полупространство H_i^+ является рецессивным относительно вектора c. *Целевой проекцией* $\gamma_i(x)$ точки $x \in \mathbb{R}^n$ на рецессивное полупространство H_i^+ называется точка, определяемая формулой

$$\gamma_i(x) = x - \sigma_i(x)c, \tag{21}$$

где

$$\sigma_i(x) = \min \left\{\sigma \in \mathbb{R}_{\geqslant 0} \;\middle|\; x - \sigma c \in H_i^+\right\} .$$

Примеры целевых проекций в $\mathbb{R}^2$ приведены на рис. 1.

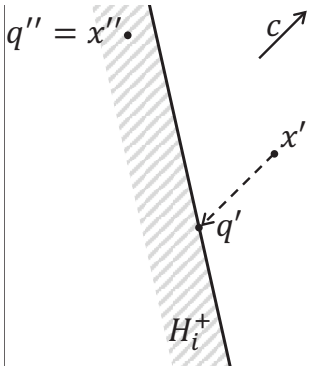

Рис. 1. Целевые проекции в пространстве $\mathbb{R}^2$: $\gamma_i(x') = q'$; $\gamma_i(x'') = q'' = x''$

Следующее утверждение предоставляет формулу для вычисления целевой проекции на полупространство, рецессивное относительно вектора c.

Утверждение 3. Пусть полупространство H_i^+, определяемое неравенством

$$\langle \tilde{a}_i, x\rangle \leqslant b_i, \tag{22}$$

является рецессивным относительно вектора c. Пусть

$$g \notin H_i^+. \tag{23}$$

Тогда

$$\gamma_i(g) = g - \frac{\langle \tilde{a}_i, g\rangle - b_i}{\langle \tilde{a}_i, c\rangle} c. \tag{24}$$

Доказательство. В соответствии с определением 4 имеем

$$\gamma_i(g) = g - \sigma_i(g)c,$$

где

$$\sigma_i(x) = \min\left\{\sigma \in \mathbb{R}_{\geqslant 0} \;\middle|\; x - \sigma c \in H_i^+\right\}.$$

Таким образом, нам необходимо показать, что

$$\frac{\langle \tilde{a}_i, g\rangle - b_i}{\langle \tilde{a}_i, c\rangle} = \min\left\{\sigma \in \mathbb{R}_{\geqslant 0} \;\middle|\; x - \sigma c \in H_i^+\right\}. \tag{25}$$

Рассмотрим прямую L, задаваемую параметрическим уравнением

$$L = \{g + \tau c | \tau \in \mathbb{R}\}.$$

Пусть точка q является пересечением прямой L с гиперплоскостью H_i:

$$q = L \cap H_i. \tag{26}$$

Тогда q должна удовлетворять уравнению

$$q = g + \tau' c \tag{27}$$

при некотором $\tau' \in \mathbb{R}$. Подставим правую часть уравнения (27) в уравнение (6) вместо x:

$$\left\langle \tilde{a}_i, g + \tau' c\right\rangle = b_i.$$

Отсюда

$$\begin{aligned} &\langle \tilde{a}_i, g\rangle + \tau' \langle \tilde{a}_i, c\rangle = b_i, \\ &\tau' = \frac{b_i - \langle \tilde{a}_i, g\rangle}{\langle \tilde{a}_i, c\rangle}. \end{aligned} \tag{28}$$

Подставив вместо τ' правую часть уравнения (28) в формулу (27) получаем

$$q = g + \frac{b_i - \langle \tilde{a}_i, g\rangle}{\langle \tilde{a}_i, c\rangle} c,$$

что эквивалентно

$$q = g - \frac{\langle \tilde{a}_i, g\rangle - b_i}{\langle \tilde{a}_i, c\rangle} c. \tag{29}$$

Так как $q \in H_i$ в соответствии с (26), формула (25) будет иметь место, если мы покажем, что

$$\forall \sigma \in \mathbb{R}_{>0} : \sigma < \frac{\langle \tilde{a}_i, g\rangle - b_i}{\langle \tilde{a}_i, c\rangle} \Rightarrow g - \sigma c \notin H_i^+. \tag{30}$$

Предположим противное, то есть существует $\sigma' > 0$ такое, что

$$\sigma' < \frac{\langle \tilde{a}_i, g\rangle - b_i}{\langle \tilde{a}_i, c\rangle} \tag{31}$$

и

$$g - \sigma' c \in H_i^+. \tag{32}$$

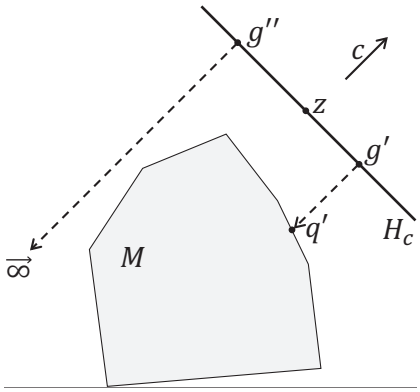

Рис. 2. Целевые проекции на многогранник M в пространстве $\mathbb{R}^2$: $\gamma_M(g') = q'$; $\gamma_M(g'') = \vec{\infty}$

Тогда из (22) и (32) следует

$$\left\langle \tilde{a}_i, g - \sigma' c \right\rangle \leqslant b_i,$$

что равносильно

$$\langle \tilde{a}_i, g \rangle - b_i \leqslant \sigma' \langle \tilde{a}_i, c \rangle . \tag{33}$$

В силу утверждения 1 имеем $\langle \tilde{a}_i, c \rangle > 0$. Поэтому (33) равносильно

$$\frac{\langle \tilde{a}_i, g \rangle - b_i}{\langle \tilde{a}_i, c \rangle} \leqslant \sigma'.$$

Получили противоречие с (31). □

Определение 5. Пусть $g \in H_c$. *Целевой проекцией* $\gamma_M(g)$ точки g на многогранник M называется точка, определяемая формулой

$$\gamma_M(g) = g - \sigma_M(g)c, \tag{34}$$

где

$$\sigma_M(g) = \min \{\sigma \in \mathbb{R}_{\geqslant 0} | g - \sigma c \in M\} .$$

Если

$$\neg\exists\, \sigma \in \mathbb{R}_{\geqslant 0} : g - \sigma c \in M,$$

полагаем $\gamma_M(g) = \vec{\infty}$ — точка, бесконечно удаленная от M.

Примеры целевых проекций на многогранник M в $\mathbb{R}^2$ приведены на рис. 2.

Определение 6. *Рецептивным полем* $\mathfrak{G}(z, \eta, \delta) \subset H_c$ плотности $\delta \in \mathbb{R}_{>0}$ с центром в точке $z \in H_c$ и рангом $\eta \in \mathbb{N}$ будем называть конечное упорядоченное множество точек, удовлетворяющих следующим условиям:

$$z \in \mathfrak{G}(z, \eta, \delta); \tag{35}$$

$$\forall g \in \mathfrak{G}(z, \eta, \delta) : \|g - z\| \leqslant \eta\delta\sqrt{n}; \tag{36}$$

$$\forall g', g'' \in \mathfrak{G}(z, \eta, \delta) : g' \neq g'' \Rightarrow \|g' - g''\| \geqslant \delta; \tag{37}$$

$$\forall g' \in \mathfrak{G}(z, \eta, \delta)\ \exists g'' \in \mathfrak{G}(z, \eta, \delta) : \|g' - g''\| = \delta; \tag{38}$$

$$\forall x \in \mathrm{Co}(\mathfrak{G}(z, \eta, \delta))\ \exists g \in \mathfrak{G}(z, \eta, \delta) : \|g - x\| \leqslant \tfrac{1}{2}\delta\sqrt{n}. \tag{39}$$

Здесь $\mathrm{Co}(X)$ обозначает выпуклую оболочку конечного множества точек $X = \left\{x^{(1)}, \ldots, x^{(K)}\right\} \subset \mathbb{R}^n$:

$$Co(X) = \left\{ \sum_{i=1}^{K} \lambda_i x^{(i)} \middle| \lambda_i \in \mathbb{R}_{\geqslant 0}, \sum_{i=1}^{K} \lambda_i = 1 \right\}.$$

Точки рецептивного поля будем называть *рецептивными точками.*

Формула (35) в определении 6 означает, что точка, находящаяся в центре рецептивного поля, принадлежит этому полю. Из формулы (36) следует, что любая точка g рецептивного поля отстоит от центральной точки z на расстояние не более $\eta\delta\sqrt{n}$. В соответствии с формулой (37) для любых двух различных точек $g' \neq g''$ рецептивного поля расстояние между ними не может быть меньше δ. Формула (38) говорит, что для любой точки g' рецептивного поля найдется точка g'', принадлежащая этому же полю, такая, что расстояние между g' и g'' будет равно δ. Формула (39) означает, что для любой точки x, принадлежащей выпуклой оболочке рецептивного поля, в этом поле найдется точка g, отстоящая от x на расстояние не более $\frac{1}{2}\delta\sqrt{n}$. Пример рецептивного поля в пространстве $\mathbb{R}^3$ приведен на рис. 3.

Опишем конструктивный метод построения рецептивного поля. Без ограничения общности мы можем предполагать, что $c_n \neq 0$. Построим в $\mathbb{R}^n$ следующий ортогональный базис, включающий в себя вектор c:

$$\begin{aligned}
&c^{(0)} = c = (c_1, c_2, c_3, c_4, \ldots, c_{n-1}, c_n);\\
&c^{(1)} = \begin{cases} \left(-\frac{1}{c_1}\sum_{i=2}^{n} c_i^2, c_2, c_3, c_4, \ldots, c_{n-1}, c_n\right), \text{ если } c_1 \neq 0; \\ (1, 0, \ldots, 0), \text{ если } c_1 = 0; \end{cases}\\
&c^{(2)} = \begin{cases} \left(0, -\frac{1}{c_2}\sum_{i=3}^{n} c_i^2, c_3, c_4, \ldots, c_{n-1}, c_n\right), \text{ если } c_2 \neq 0; \\ (0, 1, 0, \ldots, 0), \text{ если } c_2 = 0; \end{cases}\\
&c^{(3)} = \begin{cases} \left(0, 0, -\frac{1}{c_3}\sum_{i=4}^{n} c_i^2, c_4, \ldots, c_{n-1}, c_n\right), \text{ если } c_3 \neq 0; \\ (0, 0, 1, 0, \ldots, 0), \text{ если } c_3 = 0; \end{cases}\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&c^{(n-2)} = \begin{cases} \left(0, \ldots, 0, -\frac{1}{c_{n-2}}\sum_{i=n-1}^{n} c_i^2, c_{n-1}, c_n\right), \text{ если } c_{n-2} \neq 0; \\ (0, \ldots, 0, 1, 0, 0), \text{ если } c_{n-2} = 0; \end{cases}\\
&c^{(n-1)} = \begin{cases} \left(0, \ldots, 0, -\frac{c_n^2}{c_{n-1}}, c_n\right), \text{ если } c_{n-1} \neq 0; \\ (0, \ldots, 0, 0, 1, 0), \text{ если } c_{n-1} = 0. \end{cases}
\end{aligned}$$

Непосредственно проверяется, что

$$\forall i, j \in \{0, 1, \ldots, n-1\}, i \neq j : \left\langle c^{(i)}, c^{(j)} \right\rangle = 0.$$

В частности

$$\forall i = 1, \ldots, n-1 : \left\langle c, c^{(i)} \right\rangle = 0. \tag{40}$$

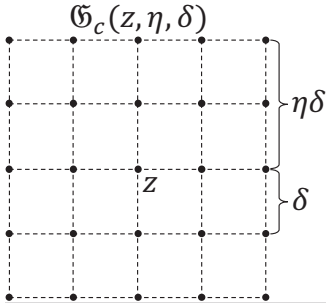

Рис. 3. Рецептивное поле в пространстве $\mathbb{R}^3$

Следующее утверждение показывает, что линейное подпространство размерности $(n-1)$, порождаемое ортогональными векторами $c_1, \ldots, c_{n-1}$, является гиперплоскостью, параллельной гиперплоскости H_c.

Утверждение 4. Определим линейное подпространство $S_c \subset \mathbb{R}^n$ размерности $n-1$:

$$S_c = \left\{ \sum_{i=1}^{n-1} \lambda_i c^{(i)} \middle| \lambda_i \in \mathbb{R} \right\}. \tag{41}$$

Тогда

$$\forall s \in S_c : s + z \in H_c. \tag{42}$$

Доказательство. Пусть $s \in S_c$, то есть

$$s = \lambda_1 c^{(1)} + \ldots + \lambda_{n-1} c^{(n-1)}.$$

Тогда

$$\langle c, (s+z) - z \rangle = \lambda_1 \left\langle c, c^{(1)} \right\rangle + \ldots + \lambda_{n-1} \left\langle c, c^{(n-1)} \right\rangle.$$

Отсюда в силу (40) получаем

$$\langle c, (s+z) - z \rangle = 0.$$

Сопоставляя это с (16), имеем $s + z \in H_c$. □

Определим векторы

$$e^{(i)} = \frac{c^{(i)}}{\|c^{(i)}\|} \ (i = 1, \ldots, n-1), \tag{43}$$

образующие ортонормированный базис подпространства S_c.

Процедура построения рецептивного поля представлена в виде алгоритма 1. Этот алгоритм строит рецептивное поле $\mathfrak{G}(z, \eta, \delta)$, состоящее из

$$K_{\mathfrak{G}} = (2\eta + 1)^{n-1} \tag{44}$$

точек, расположенных в узлах регулярной решетки, имеющей форму гиперквадрата (гиперкуба размерности $n-1$) с длиной ребра, равной $2\eta\delta$. Длина ребра ячейки гиперквадрата равна δ. В соответствии с шагом 13 алгоритма 1 и утверждением 4 этот гиперквадрат лежит в гиперплоскости H_c и имеет центр в точке z. Недостаток алгоритма 1 состоит в том, что количество вложенных циклов **for** зависит от размерности пространства. Эту проблему можно решить с помощью функции G, вычисляющей точку рецептивного множества по ее

Алгоритм 1 Построение рецептивного поля $\mathfrak{G}(z, \eta, \delta)$

Require: $z \in H_c$, $\eta \in \mathbb{N}$, $\delta \in \mathbb{R}_{>0}$

$\mathfrak{G} := \emptyset$
for $i_{n-1} = 0 \dots 2\eta$ **do**
  $s_{n-1} := i_{n-1}\delta - \eta\delta$
  for $i_{n-2} = 0 \dots 2\eta$ **do**
    $s_{n-2} := i_{n-2}\delta - \eta\delta$
    $\dots$
    for $i_1 = 0 \dots 2\eta$ **do**
      $s_1 := i_1\delta - \eta\delta$
      $s := \mathbf{0}$
      for $j = 1 \dots n-1$ **do**
        $s := s + s_j e^{(j)}$
      end for
      $\mathfrak{G} := \mathfrak{G} \cup \{s + z\}$
    end for
  end for
end for

Алгоритм 2 Функция G вычисляет точку рецептивного поля по ее номеру k

Require: $z \in H_c$, $\eta \in \mathbb{N}$, $\delta \in \mathbb{R}_{>0}$

function $\mathrm{G}(k, n, z, \eta, \delta)$
  for $j = (n-1) \dots 1$ **do**
    $l_j := \lfloor k/(2\eta+1)^{j-1} \rfloor$
    $k := k \mod (2\eta+1)^{j-1}$
  end for
  $g := z$
  for $j = 1 \dots (n-1)$ **do**
    $g := g + (l_j\delta - \eta\delta)e^{(j)}$
  end for
  $\mathrm{G} := g$
end function

порядковому номеру (нумерация начинается с нуля; порядок определяется алгоритмом 1). Реализация функции G представлена в виде алгоритма 2. Следующее утверждение 5 дает оценку временной сложности алгоритма 2.

Утверждение 5. Алгоритм 2 допускает реализацию с временной сложностью[1]

$$c_G = 4n^2 + 5n - 9, \tag{45}$$

[1]Под временной сложностью здесь понимается количество арифметических операций и операций сравнения, необходимых для выполнения алгоритма.

где n — размерность пространства.

Доказательство. Рассмотрим низкоуровневую реализацию алгоритма 2, представленную в виде алгоритма 3.

Алгоритм 3 Низкоуровневая реализация алгоритма 2

1: $p := 2\eta + 1$; $r := \eta\delta$; $h := p^{n-2}$; $g := z$
2: $j := n - 1$
3: **repeat**
4: $\quad l_j := \lfloor k/h \rfloor$
5: $\quad k := k \mod h$
6: $\quad h := h/p$
7: $\quad j := j - 1$
8: **until** $j = 0$
9: $j := 1$
10: **repeat**
11: $\quad w_j := l_j\delta - r$
12: $\quad i := 1$
13: $\quad$ **repeat**
14: $\quad\quad g_i := g_i + w_j e_i^{(j)}$
15: $\quad\quad i := i + 1$
16: $\quad$ **until** $i > n$
17: $\quad j := j + 1$
18: **until** $j = n$

Значения, вычисляемые на шагах 1–2 алгоритма 3, не зависят от номера рецептивной точки k и поэтому могут считаться константами. Цикл **repeat**/**until** на шагах 3–8 выполняется $(n-1)$ раз и требует $c_{3:8} = 5(n-1)$ операций. Вложенный цикл **repeat**/**until** на шагах 13–16 выполняется n раз и требует $c_{13:16} = 4n$ операций. Внешний цикл **repeat**/**until** на шагах 10–18 выполняется $(n-1)$ раз и требует $c_{10:18} = (4 + c_{13-16})(n-1) = 4(n^2-1)$ операций. В сумме получаем

$$c_G = c_{3:8} + c_{10:18} = 4n^2 + 5n - 9.$$

□

Следствие 1. Временная сложность алгоритма 2 может быть оценена как $O(n^2)$.

Определение 7. Пусть $z \in H_c$. Зафиксируем $\eta \in \mathbb{N}$, $\delta \in \mathbb{R}_{>0}$. *Образом* $\mathfrak{I}(z, \eta, \delta)$, порожденным рецептивным полем $\mathfrak{G}(z, \eta, \delta)$, будем называть упорядоченное множество вещественных чисел

$$\mathfrak{I}(z, \eta, \delta) = \{\rho_c(\gamma_M(g)) | g \in \mathfrak{G}(z, \eta, \delta)\}. \tag{46}$$

Порядок чисел в образе определяется порядком соответствующих точек рецептивного поля.

Функцию построения образа $\mathfrak{I}(z, \eta, \delta)$ в виде списка чисел представлена в виде алгоритма 4. Здесь [] обозначает пустой список, а $+\!\!+$ обозначает операцию конкатенации списков.

Алгоритм 4 Построение образа $\mathfrak{I}(z, \eta, \delta)$

Require: $z \in H_c$, $\eta \in \mathbb{N}$, $\delta \in \mathbb{R}_{>0}$

function $\mathfrak{I}(z, \eta, \delta)$
  $\mathfrak{I} := [\,]$
  for $k = 0 \ldots ((2\eta+1)^{n-1} - 1)$ **do**
    $g_k := \mathrm{G}(k, n, z, \eta, \delta)$
    $\mathfrak{I} := \mathfrak{I} +\!\!+ [\rho_c(\gamma_M(g_k))]$
  end for
end function

Покажем как полученный образ может быть использован для решения задачи ЛП. Пусть $\langle \tilde{a}_i, c\rangle > 0$. Это означает, что полупространство H_i^+ является рецессивным относительно вектора c (см. утверждение 1). Пусть имеется точка $u \in H_i \cap M$. Допустим, что нам удалось создать *искусственную нейронную сеть* DNN, получающую на входе образ $\mathfrak{I}(\pi_c(u), \eta, \delta)$ окрестности точки u, и выдающую на выходе точку u' такую, что

$$u' = \arg\min \{\rho_c(x) | x \in H_i \cap M\}.$$

Тогда мы можем построить алгоритм 5, решающий задачу линейного программирования (20) с помощью DNN.

Алгоритм 5 Линейное программирование с использованием DNN

Require: $u^{(1)} \in H_i \cap M$, $\langle \tilde{a}_i, c\rangle > 0$, $z \in H_c$; $\eta \in \mathbb{N}$, $\delta \in \mathbb{R}_{>0}$

$k := 1$
repeat
  $\mathcal{I} := \mathfrak{I}(u^{(k)}, \eta, \delta)$
  $u^{(k+1)} := \mathrm{DNN}(\mathcal{I})$
  $k := k + 1$
until $u^{(k)} \neq u^{(k-1)}$
$\bar{x} := u^{(k)}$
stop

2. Параллельный алгоритм построения образа задачи ЛП

При решении задач ЛП большой размерности и с большим количеством ограничений алгоритм 4 построения образа задачи ЛП может потребовать значительных временных затрат. В этом разделе приводится его параллельная версия, позволяющая существенно сократить время решения задачи ЛП с помощью алгоритма 5. Параллельная версия алгоритма 4 строится на основе модели параллельных вычислений BSF [24, 25], ориентированной на кластерные вычислительные системы. Модель BSF использует парадигму мастер–рабочие и требует представление алгоритма в форме операций над списками с использованием функций высшего порядка *Map* и *Reduce*, определенных в формализме Бёрда—Миртенса (Bird–Meertens formalism) [26]. Модель BSF также предоставляет метрику для аналитиче-

ской оценки масштабируемости параллельного алгоритма, удовлетворяющего указанным требованиям.

Представим алгоритм 4 в форме операций над списками с использованием функций высшего порядка *Map* и *Reduce*. В качестве списка, обрабатываемого функцией высшего порядка *Map*, возьмем список номеров неравенств системы (4):

$$\mathcal{L}_{map} = [1, \ldots, m]\,. \tag{47}$$

Обозначим $\mathbb{R}_\infty = \mathbb{R} \cup \{\infty\}$. Определим следующим образом параметризованную функцию $\mathrm{F}_k : \{1, \ldots, m\} \to \mathbb{R}_\infty$, являющуюся первым параметром функции высшего порядка *Map*:

$$\mathrm{F}_k(i) = \begin{cases} \rho_c\left(\gamma_i(g_k)\right), \text{ если } \langle \tilde{a}_i, c\rangle > 0 \text{ и } \gamma_i(g_k) \in M; \\ \infty, \text{ если } \langle \tilde{a}_i, c\rangle \leqslant 0 \text{ или } \gamma_i(g_k) \notin M, \end{cases} \tag{48}$$

где $g_k = G(k, n, z, \eta, \delta)$ вычисляется с помощью алгоритма 2, а $\gamma_i(g_k)$ вычисляется по формуле (24). С неформальной точки зрения функция F_k отображает номер полупространства H_i^+ в расстояние от целевой проекции до целевой гиперплоскости, если H_i^+ является рецессивным относительно c (см. утверждение 1) и целевая проекция принадлежит M. В противном случае F_k возвращает специальное значение ∞.

Функция высшего порядка *Map* преобразует список $\mathcal{L}_{map}$ в список $\mathcal{L}_{reduce}$ путем применения функции F_k к каждому элементу списка $\mathcal{L}_{map}$:

$$\mathcal{L}_{reduce} = \mathrm{Map}\left(\mathrm{F}_k, \mathcal{L}_{map}\right) = \left[\mathrm{F}_k(1), \ldots, \mathrm{F}_k(m)\right] = [\rho_1, \ldots, \rho_m]\,.$$

Определим бинарную ассоциативную операцию $\Downarrow : \mathbb{R}_\infty \to \mathbb{R}_\infty$ следующим образом:

$$\begin{aligned} \infty \Downarrow \infty &= \infty; \\ \forall \alpha \in \mathbb{R} : \alpha \Downarrow \infty &= \alpha; \\ \forall \alpha, \beta \in \mathbb{R} : \alpha \Downarrow \beta &= \min(\alpha, \beta). \end{aligned}$$

С неформальной точки зрения операция $\Downarrow$ вычисляет минимальное из двух чисел.

Функция высшего порядка *Reduce* преобразует список $\mathcal{L}_{reduce}$ в атомарное значение $\rho \in \mathbb{R}_\infty$ путем последовательного применения операции $\Downarrow$ ко всему списку:

$$\mathrm{Reduce}(\Downarrow, \mathcal{L}_{reduce}) = \rho_1 \Downarrow \rho_2 \Downarrow \ldots \Downarrow \rho_m = \rho.$$

Построение образа $\mathfrak{I}$ задачи ЛП с использованием функций высшего порядка *Map* и *Reduce* представлено в виде алгоритма 6. Параллельная версия алгоритма 6 строится на основе алгоритмического шаблона 2 из [24]. Результат представлен в виде алгоритма 7. Прокомментируем параллельный алгоритм 7. Для простоты мы будем предполагать, что количество ограничений m кратно количеству рабочих L и нумерация неравенств начинается с нуля. Параллельный алгоритм включает в себя $L + 1$ процесс: один процесс-мастер (кратко — мастер) и L процессов-рабочих (кратко — рабочие). Мастер управляет вычислениями. Первоначально в качестве образа $\mathfrak{I}$ берется пустой список (шаг 2 мастера). Текущий номер k полагается равным нулю (шаг 3 мастера). На шагах 4–15 мастер организует цикл **repeat**/**until**, в котором строится образ $\mathfrak{I}$ задачи ЛП. На шаге 5 мастер посылает номер очередной рецептивной точки k всем рабочим. На шаге 8 мастер ожидает получение

Алгоритм 6 Построение образа $\mathfrak{I}$ с использованием *Map* и *Reduce*

Require: $z \in H_c$, $\eta \in \mathbb{N}$, $\delta \in \mathbb{R}_{>0}$

1: **input** $n, m, A, b, c, z, \eta, \delta$
2: $\mathfrak{I} := [\,]$
3: $\mathcal{L}_{map} := [1, \ldots, m]$
4: **for** $k = 0 \ldots ((2\eta+1)^{n-1} - 1)$ **do**
5: $\quad \mathcal{L}_{reduce} := \mathrm{Map}(\mathrm{F}_k, \mathcal{L}_{map})$
6: $\quad \rho := \mathrm{Reduce}(\Downarrow, \mathcal{L}_{reduce})$
7: $\quad \mathfrak{I} := \mathfrak{I} +\!\!+ [\rho]$
8: **end for**
9: **output** $\mathfrak{I}$
10: **stop**

Алгоритм 7 Параллельный алгоритм построение образа $\mathfrak{I}$ задачи ЛП

Мастер	**Рабочий (l=0,. . . ,L-1)**
1: **input** n	1: **input** $n, m, A, b, c, z, \eta, \delta$
2: $\mathfrak{I} := [\,]$	2: $L := \mathrm{NumberOfWorkers}$
3: $k := 0$	3: $\mathcal{L}_{map(l)} := [lm/L, \ldots, ((l+1)m/L) - 1]$
4: **repeat**	4: **repeat**
5: $\quad$ **SendToWorkers** k	5: $\quad$ **RecvFromMaster** k
6:	6: $\quad \mathcal{L}_{reduce(l)} := \mathrm{Map}\left(\mathrm{F}_k, \mathcal{L}_{map(l)}\right)$
7:	7: $\quad \rho_l := \mathrm{Reduce}\left(\Downarrow, \mathcal{L}_{reduce(l)}\right)$
8: $\quad$ **RecvFromWorkers** $[\rho_0, \ldots, \rho_{L-1}]$	8: $\quad$ **SendToMaster** ρ_l
9: $\quad \rho := \mathrm{Reduce}\,(\Downarrow, [\rho_0, \ldots, \rho_{L-1}])$	9:
10: $\quad \mathfrak{I} := \mathfrak{I} +\!\!+ [\rho]$	10:
11: $\quad k := k + 1$	11:
12: $\quad exit := \left(k \geqslant (2\eta+1)^{n-1}\right)$	12:
13: $\quad$ **SendToWorkers** $exit$	13: $\quad$ **RecvFromMaster** $exit$
14: **until** $exit$	14: **until** $exit$
15: **output** $\mathfrak{I}$	15:
16: **stop**	16: **stop**

частичных результатов от всех рабочих. Эти результаты редуцируются в одно значение, которое добавляется в образ $\mathfrak{I}$ (шаги 9–10 мастера). Шаг 11 увеличивает счетчик итераций k на единицу. На шаге 12 мастер вычисляет критерий завершения в виде логического выражения $\left(k \geqslant (2\eta+1)^{n-1}\right)$, значение которого присваивается булевой переменной *exit*. В соответствии с формулой (44) значение true будет означать, что точки рецептного поля закончились. На шаге 13 мастер посылает значение булевой переменной *exit* всем рабочим. Если обработаны не все точки рецептивного поля, то на шаге 14 происходит переход к выполнению следующей итерации цикла **repeat**/**until**. В противном случае мастер выводит построенный образ $\mathfrak{I}$ (шаг 15) и завершает свою работу (шаг 16).

Каждый l-тый рабочий выполняет общую последовательность действий, но над своей частью $\mathcal{L}_{map(l)}$ списка $\mathcal{L}_{map}$, которая определяется на шаге 3. На шаге 4 рабочий входит в цикл **repeat/until**. На шаге 5 он получает номер очередной точки k, принадлежащей рецептивному полю. На шаге 6 рабочий обрабатывает свой подсписок $\mathcal{L}_{map(l)}$, используя функцию высшего порядка *Map*, которая для каждого элемента подсписка выполняет параметризованную функцию F_k, определенную с помощью формулы (48). В результате получается подсписок $\mathcal{L}_{reduce(l)}$, содержащий расстояния $\mathrm{F}_k(i)$ от целевой гиперплоскости H_c до целевых проекций рецептивной точки g_k на гиперплоскости H_i для всех i из подсписка $\mathcal{L}_{map(l)}$. На шаге 7 рабочий с помощью функции высшего порядка *Reduce* редуцирует подсписок $\mathcal{L}_{reduce(l)}$ в атомарное значение ρ_l, используя ассоциативную бинарную операцию $\textcircled{\downarrow}$, вычисляющую минимальное расстояние. Полученный частный результат пересылается мастеру (шаг 8 рабочего). На шаге 13 рабочий ожидает получение от мастера значения булевой переменной *exit*. Если получено значение false, рабочий продолжает выполнение цикла **repeat/until** (шаг 14 рабочего). В противном случае процесс рабочего завершается на шаге 16.

Дадим *аналитическую оценку границы масштабируемости* параллельного алгоритма 7 с использованием стоимостной метрики модели параллельных вычислений BSF [24]. Под границей масштабируемости здесь понимается число рабочих, на котором достигается максимум ускорения. Стоимостная метрика модели BSF включает в себя следующие стоимостные параметры для цикла **repeat/until** (шаги 4–14) параллельного алгоритма 7:

- m : длина списка $\mathcal{L}_{map}$;
- D : латентность (время пересылки одно байта от мастера к рабочему);
- t_c : время, затрачиваемое мастером на пересылку одному рабочему координат рецептивной точки и получения от него расстояния от этой точки до целевой проекции (включая латентность);
- t_{Map} : время, затрачиваемое одним рабочим на выполнение функции высшего порядка *Map* для всего списка $\mathcal{L}_{map}$;
- t_a : время, затрачиваемое на выполнение одной бинарной операции $\textcircled{\downarrow}$.

В соответствии с формулой (14) из [24] граница масштабируемости параллельного алгоритма 7 может быть оценена следующим образом:

$$L_{max} = \frac{1}{2}\sqrt{\left(\frac{t_c}{t_a \ln 2}\right)^2 + \frac{t_{Map}}{t_a} + 4m} - \frac{t_c}{t_a \ln 2}. \tag{49}$$

Вычислим оценку для временных параметров формулы (49). Для этого ведем следующие обозначения в рамках одной итерации цикла **repeat/until** (шаги 4–14) параллельного алгоритма 7:

- c_c : количество числе, пересылаемых от мастера к рабочему и обратно в рамках одной итерации;
- c_{Map} : количество арифметических операций и операций сравнения, выполняемых на шаге 5 последовательного алгоритма 6;
- c_a : количество операций, необходимых для выполнения бинарной операции $\textcircled{\downarrow}$.

В начале каждой итерации мастер посылает каждому рабочему номер рецептивной точки. В ответ рабочий посылает расстояние от этой точки до целевой проекции. Следовательно

$$c_c = 2. \tag{50}$$

В контексте алгоритма 6

$$c_{Map} = (c_G + c_{F_k})\, m, \tag{51}$$

где c_G — количество операций, необходимых для вычисления координат точки g_k, c_{F_k} — количество операций, необходимых для вычисления функции $\mathrm{F}_k(i)$ по формуле (48) в предположении, что координаты точки g_k уже вычислены. Оценка значения c_G дана в предложении 5. Оценим значение c_{F_k}. В соответствии с (24) вычисление целевой проекции $\gamma_i(g_k)$ требует $(6n-2)$ арифметических операций. Из (19) следует, что вычисление $\rho_c(x)$ составляет $(5n-1)$ арифметических операций. Проверка условия $x \in M$ в соответствии с (4) потребует $m(2n-1)$ арифметических операций и m операций сравнения. Таким образом

$$c_{F_k} = 2mn + 11n - 3. \tag{52}$$

Подставив в (51) правые части формул (45) и (52), получаем

$$c_{Map} = 4n^2m + 2m^2n + 16nm - 12m. \tag{53}$$

Для выполнения бинарной операции $\textcircled{\downarrow}$ необходимо выполнить одну операцию сравнения:

$$c_a = 1. \tag{54}$$

Пусть τ_{op} — среднее время выполнения арифметических операций и операций сравнения, τ_{tr} — среднее время для пересылки одного вещественного числа без учета латентности. Тогда, используя формулы (50), (53) и (54), получаем

$$t_c = c_c\tau_{tr} + 2D = 2(\tau_{tr} + D); \tag{55}$$

$$t_{Map} = c_{Map}\tau_{op} = (4n^2m + 2m^2n + 16nm - 12m)\tau_{op}; \tag{56}$$

$$t_a = c_a\tau_{op} = \tau_{op}. \tag{57}$$

Подставив в (49) правые части формул (55)–(57), получим следующую оценку границы масштабируемости параллельного алгоритма 7:

$$L_{max} = \frac{1}{2}\sqrt{\left(\frac{2(\tau_{tr}+D)}{\tau_{op}\ln 2}\right)^2 + 4n^2m + 2m^2n + 16nm - 12m} - \frac{2(\tau_{tr}+D)}{\tau_{op}\ln 2}.$$

Для больших значений m и n это эквивалентно

$$L_{max} \approx O(\sqrt{2n^2m + m^2n + 8nm - 6m}). \tag{58}$$

Если предположить, что $m = O(n)$, то из (58) следует

$$L_{max} \approx O(n\sqrt{n}), \tag{59}$$

Таблица 1. Характеристики кластера «Торнадо ЮУрГУ»

Параметр	Значение
Количество процессорных узлов	480
Процессоры	Intel Xeon X5680 (6 ядер, 3.33 GHz)
Количество процессоров в узле	2
Оперативная память узла	24 GB DDR3
Соединительная сеть	InfiniBand QDR (40 Gbit/s)
Операционная система	Linux CentOS

где n — размерность пространства. Оценка (59) позволяет сделать вывод, что параллельный алгоритм 7 демонстрирует превосходную масштабируемость[2]. В следующем разделе мы проверим аналитическую оценку (59) путем проведения масштабных вычислительных экспериментов на реальной кластерной вычислительной системе.

3. Вычислительные эксперименты

Нами была выполнена параллельная реализация алгоритма 7 в виде программы ViLiPP (Visualization of Linear Programming Problem) на языке C++ с использованием программного BSF-каркаса [27, 28]. BSF-каркас базируется на модели параллельных вычислений BSF и инкапсулирует все аспекты, связанные с распараллеливанием программы с использованием библиотеки MPI [29] и программного интерфейса OpenMP [30]. Исходные коды программы ViLiPP свободно доступны в сети Интернет по адресу `https://github.com/nikolay-olkhovskiy/LP-visualization-MPI`. С использованием параллельной программы ViLiPP мы провели эксперименты по исследованию масштабируемости алгоритма 7 на кластерной вычислительной системе «Торнадо ЮУрГУ» [31], характеристики которой приведены в таблице 1.

С помощью генератора задач «FRaGenLP» [32, 33] для проведения вычислительных экспериментов были сгенерированы три случайные задачи ЛП, параметры которых приведены в таблице 2. Количество ненулевых значений матрицы A задачи (1) всех случаях составило 100%. Для всех задач ранг рецептивного поля η полагался равным 2. В соответствии с формулой (44) мощность рецептивного поля демонстрировала экспоненциальный рост с увеличением размерности пространства.

Результаты вычислительных экспериментов приведены в таблице 3 и на рис. 4. Во всех запусках каждому рабочему выделялся отдельный процессорный узел. Еще один дополнительный процессорный узел выделялся для работы мастера. Вычислительные эксперименты показали, что с ростом размерности задачи наблюдается рост границы масштабируемости программы ViLiPP: для LP5 максимум кривой ускорения достигается в районе 190 узлов, для LP6 максимум располагается в районе 260 узлов, а для LP7 он приблизительно равен 326 узлам. При этом наблюдается экспоненциальный рост времени решения задачи: образ задачи LP5 на 11 процессорных узлах строится за 10 сек., а построение образа задачи LP7 на таком же количестве узлов требует уже 5 мин. Дополнительный вычислительный

[2]Пусть $L_{max} = O(n^{\alpha})$. Алгоритм демонстрирует *превосходную масштабируемость*, если $\alpha > 1$; алгоритм демонстрирует *хорошую масштабируемость*, если $\alpha = 1$; алгоритм демонстрирует *ограниченную масштабируемость*, если $0 < \alpha < 1$; алгоритм *не масштабируется*, если $\alpha = 0$.

Таблица 2. Параметры тестовых задач ЛП

Идентификатор задачи	Число переменных	Число ограничений	Процент ненулевых значений в A	Мощность рецептивного поля
$LP7$	7	4016	100%	15 625
$LP6$	6	4014	100%	3 125
$LP5$	5	4012	100%	625

Таблица 3. Время построения образа задач ЛП (сек.)

Число процессорных узлов	LP5	LP6	LP7
11	9.81	54.45	303.78
56	1.93	10.02	59.43
101	1.55	6.29	33.82
146	1.39	4.84	24.73
191	1.35	4.20	21.10
236	1.38	3.98	19.20
281	1.45	3.98	18.47
326	1.55	4.14	18.30

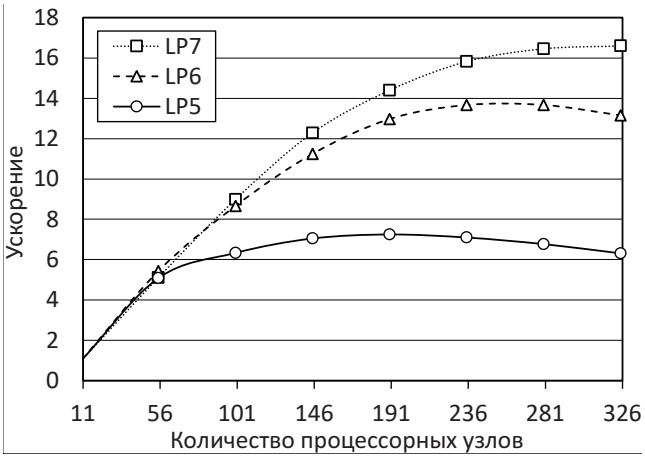

Рис. 4. Графики ускорения параллельной программы ViLiPP для задач ЛП различного размера

эксперимент показал, что построение образа задачи при $n = 9$ на 11 процессорных узлах занимает 1.5 часа.

Проведенные эксперименты позволяют сделать вывод, что при современном уровне развития вычислительной техники применение искусственных нейронных сетей для решения задач ЛП на основе предложенного метода визуализации может быть эффективным для задач размерности, не превышающей 100, с количеством ограничений до 100 000.

Заключение

Основным результатом, полученным в данной работе, является математическая модель визуального представления многомерной задачи линейного программирования по поиску в допустимой области точки максимума линейной целевой функции. Основным элементом модели является рецептивное поле, представляющее собой регулярное множество точек, располагающихся в узлах решетки, построенной внутри гиперкуба. Все точки рецептивного поля лежат в целевой гиперплоскости, ортогональной вектору $c = (c_1, \ldots, c_n)$, составленному из коэффициентов линейной целевой функции. Целевая гиперплоскость располагается так, что для любой точки x из допустимой области и любой точки z целевой гиперплоскости выполняется неравенство $\langle c, x\rangle < \langle c, z\rangle$. Можно сказать, что рецептивное поле является многомерным абстрактным аналогом светочувствительной матрицы цифрового фотоаппарата. Из каждой точки рецептивного поля в направлении допустимой области строится луч, параллельный вектору c. Точка, в которой луч касается допустимой области, называется целевой проекцией. Образ задачи линейного программирования представляет собой матрицу положительных вещественных чисел размерности $(n-1)$, в которой каждый элемент является расстоянием от точки рецептивного поля до соответствующей точки целевой проекции.

В статье описан алгоритм вычисления координат точки рецептивного поля по ее порядковому номеру. Показано, что временная сложность этого алгоритма может быть оценена как $O(n^2)$, где n — размерность пространства. Приведено общее описание алгоритма решения задачи линейного программирования с помощью искусственной нейронной сети на основе анализа построенных образов. Предложен параллельный алгоритм построения образа задачи линейного программирования, ориентированный на кластерные вычислительные системы. Этот алгоритм основывается на модели параллельных вычислений BSF, предполагающей использование парадигмы мастер–рабочие и представление алгоритма в виде операций над списками с использованием функций высшего порядка *Map* и *Reduce*. Показано, что для границы масштабируемости параллельного алгоритма справедлива оценка $O(n\sqrt{n})$. Это означает, что алгоритм демонстрирует хорошую масштабируемость.

Выполнена реализация параллельного алгоритма построения образа задачи линейного программирования на языке C++ с использованием параллельного программного BSF-каркаса, инкапсулирующего все аспекты, связанные с распараллеливанием на основе библиотеки MPI и программного интерфейса OpenMP. С использованием этой программной реализации на вычислительном кластере «Торнадо ЮУрГУ» проведены масштабные вычислительные эксперименты по построению образов для случайных многомерных задач линейного программирования с большим числом ограничений. Проведенные эксперименты подтверждают корректность и эффективность предложенных подходов. Вместе с тем следует отметить, что время построение образа растет экспоненциально с увеличением размерности пространства. Поэтому предложенный метод применим для задач с числом переменных, не превышающим 100. При этом количество ограничений теоретически может быть неограниченным.

В качестве направлений дальнейших исследований можно выделить следующие.

1. Разработать метод решения задачи линейного программирования на основании анализа ее образов и доказать его сходимость.
2. Разработать и реализовать метод построения обучающих множеств для создания нейронной сети, решающей задачи линейного программирования путем анализа их образов.

3. Разработать и обучить искусственную нейронную сеть для решения многомерные задачи линейного программирования.
4. Разработать и реализовать на вычислительном кластере параллельную программу, строящую многомерные образы задачи линейного программирования и получающую ее решение с помощью искусственной нейронной сети.

Исследование выполнено при финансовой поддержке РФФИ (грант 20-07-00092 а) и Министерства науки и высшего образования РФ (государственное задание FENU-2020-0022).

Литература

1. Jagadish H.V., Gehrke J., Labrinidis A., *et al.* Big data and its technical challenges // Communications of the ACM. 2014. Vol. 57, no. 7. P. 86–94. DOI: 10.1145/2611567.
2. Hartung T. Making Big Sense From Big Data // Frontiers in Big Data. 2018. Vol. 1. P. 5. DOI: 10.3389/fdata.2018.00005.
3. Соколинская И., Соколинский Л. О решении задачи линейного программирования в эпоху больших данных // Параллельные вычислительные технологии (ПаВТ'2017). Короткие статьи и описания плакатов. Челябинск: Издательский центр ЮУрГУ, 2017. С. 471—484. URL: http://omega.sp.susu.ru/pavt2017/short/014.pdf.
4. Chung W. Applying large-scale linear programming in business analytics // 2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM). IEEE, 2015. P. 1860–1864. DOI: 10.1109/IEEM.2015.7385970.
5. Gondzio J., Gruca J.A., Hall J.A.J., *et al.* Solving large-scale optimization problems related to Bell's Theorem // Journal of Computational and Applied Mathematics. 2014. Vol. 263. P. 392–404. DOI: 10.1016/j.cam.2013.12.003.
6. Sodhi M.S. LP modeling for asset-liability management: A survey of choices and simplifications // Operations Research. 2005. Vol. 53, no. 2. P. 181–196. DOI: 10.1287/opre.1040.0185.
7. Brogaard J., Hendershott T., Riordan R. High-Frequency Trading and Price Discovery // Review of Financial Studies. 2014. Vol. 27, no. 8. P. 2267–2306. DOI: 10.1093/rfs/hhu032.
8. Дышаев М.М., Соколинская И.М. Представление торговых сигналов на основе адаптивной скользящей средней Кауфмана в виде системы линейных неравенств // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. 2014. Т. 2, № 4. С. 103—108. DOI: 10.14529/cmse130408.
9. Bixby R.E. Solving Real-World Linear Programs: A Decade and More of Progress // Operations Research. 2002. Vol. 50, no. 1. P. 3–15. DOI: 10.1287/opre.50.1.3.17780.
10. Dongarra J., Gottlieb S., Kramer W.T.C. Race to Exascale // Computing in Science and Engineering. 2019. Vol. 21, no. 1. P. 4–5. DOI: 10.1109/MCSE.2018.2882574.
11. Dantzig G.B. Linear programming and extensions. Princeton, N.J.: Princeton university press, 1998. 656 p.
12. Zadeh N. A bad network problem for the simplex method and other minimum cost flow algorithms // Mathematical Programming. 1973. Vol. 5, no. 1. P. 255–266. DOI: 10.1007/BF01580132.

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

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

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

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

17. Ершова А.В., Соколинская И.М. О сходимости масштабируемого алгоритма построения псевдопроекции на выпуклое замкнутое множество // Вестник Южно-Уральского государственного университета. Серия: Математическое моделирование и программирование. 2011. № 37(254). С. 12—21. URL: https://mmp.susu.ru/article/ru/127.

18. Hafsteinsson H., Levkovitz R., Mitra G. Solving large scale linear programming problems using an interior point method on a massively parallel SIMD computer // Parallel Algorithms and Applications. 1994. Vol. 4, no. 3-4. P. 301–316. DOI: 10.1080/10637199408915470.

19. Karypis G., Gupta A., Kumar V. A parallel formulation of interior point algorithms // Proceedings of the 1994 ACM/IEEE conference on Supercomputing (Supercomputing'94). Los Alamitos, CA, USA: IEEE Computer Society Press, 1994. P. 204–213. DOI: 10.1109/SUPERC.1994.344280.

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

21. Schmidhuber J. Deep learning in neural networks: An overview // Neural Networks. 2015. Vol. 61. P. 85–117. DOI: 10.1016/j.neunet.2014.09.003.

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

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

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

25. Ежова Н.А., Соколинский Л.Б. Модель параллельных вычислений BSF-MR // Системы управления и информационные технологии. 2019. № 3(77). С. 15—21.

26. Bird R.S. Lectures on Constructive Functional Programming // Constructive Methods in Computing Science. NATO ASI Series F: Computer and Systems Sciences, vol. 55 / ed. by M. Broy. Berlin, Heidelberg: Springer, 1988. P. 151–216.

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

28. Sokolinsky L.B. BSF-skeleton. 2019. URL: https://github.com/leonid-sokolinsky/BSF-skeleton (дата обращения: 24.01.2022).

29. Gropp W. MPI 3 and Beyond: Why MPI Is Successful and What Challenges It Faces // Recent Advances in the Message Passing Interface. EuroMPI 2012. Lecture Notes in Computer Science, vol. 7490 / ed. by J. Traff, S. Benkner, J. Dongarra. Berlin, Heidelberg: Springer, 2012. P. 1–9. DOI: 10.1007/978-3-642-33518-1_1.

30. Kale V. Shared-memory Parallel Programming with OpenMP // Parallel Computing Architectures and APIs. Boca Raton: Chapman, Hall/CRC, 2019. Chap. 14. P. 213–222. DOI: 10.1201/9781351029223-18/SHARED-MEMORY-PARALLEL-PROGRAMMING-OPENMP-VIVEK-KALE.

31. Kostenetskiy P., Semenikhina P. SUSU Supercomputer Resources for Industry and fundamental Science // Proceedings - 2018 Global Smart Industry Conference, GloSIC 2018. IEEE, 2018. Article 8570068. P. 1–7. DOI: 10.1109/GloSIC.2018.8570068.

32. Соколинский Л.Б., Соколинская И.М. О генерации случайных задач линейного программирования на кластерных вычислительных системах // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 1. С. 38—52. DOI: 10.14529/cmse210103.

33. Соколинский Л.Б. Свидетельство о государственной регистрации программы для ЭВМ № 2021619526 Российская Федерация. Генератор случайных задач линейного программирования FRaGenLP : № 2021618165 : заявл. 28.05.2021 : опубл. 10.06.2021.

Ольховский Николай Александрович, аспирант, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

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

DOI: 10.14529/cmse220103

VISUAL REPRESENTATION OF MULTIDIMENSIONAL LINEAR PROGRAMMING PROBLEMS

© 2022 N.A. Olkhovsky, L.B. Sokolinsky

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

E-mail: olkhovskiina@susu.ru, leonid.sokolinsky@susu.ru

Received: 10.03.2022

The article proposes an n-dimensional mathematical model of the visual representation of a linear programming problem. This model makes it possible to use artificial neural networks to solve multidimensional linear optimization problems, the feasible region of which is a bounded non-empty set. To visualize the linear programming problem, an objective hyperplane is introduced, the orientation of which is determined by the gradient of the linear objective function: the gradient is the normal to the objective hyperplane. In case of searching a maximum, the objective hyperplane is positioned in such a way that the value of the objective function at all its points exceeds the value of the objective function at all points of the feasible region, which is a bounded convex polytope. For an arbitrary point of the objective hyperplane, the objective projection onto the polytope is determined: the closer the objective projection point is to the objective hyperplane, the greater the value of the objective function at this point. Based on the objective hyperplane, a finite regular set of points is constructed, called the retina. Using objective projections, an image of a polytope is constructed. This image includes the points of the retina and the distances to the corresponding points of the polytope surface. Based on the proposed model, parallel algorithms for visualizing a linear programming problem are constructed. An analytical estimation of its scalability is performed. Information about the software implementation and the results of large-scale computational experiments confirming the efficiency of the proposed approaches are presented.

Keywords: linear programming, n-dimensional visualization, mathematical model, parallel algorithm, BSF-skeleton.

FOR CITATION

Olkhovsky N.A., Sokolinsky L.B. Visual Representation of Multidimensional Linear Programming Problems. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2022. Vol. 11, no. 1. P. 31–56. (in Russian) DOI: 10.14529/cmse220103.

References

1. Jagadish H.V., Gehrke J., Labrinidis A., *et al.* Big data and its technical challenges. Communications of the ACM. 2014. Vol. 57, no. 7. P. 86–94. DOI: 10.1145/2611567.
2. Hartung T. Making Big Sense From Big Data. Frontiers in Big Data. 2018. Vol. 1. P. 5. DOI: 10.3389/fdata.2018.00005.
3. Sokolinskaya I.M., Sokolinsky L.B. On solving linear programming problems in the era of big data. Parallel Computing Technologies (PCT'2017). Short articles and posters. Chelyabinsk: SUSU Publishing Center, 2017. P. 471–484. (in Russian).
4. Chung W. Applying large-scale linear programming in business analytics. 2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM). IEEE, 2015. P. 1860–1864. DOI: 10.1109/IEEM.2015.7385970.

5. Gondzio J., Gruca J.A., Hall J.A.J., *et al.* Solving large-scale optimization problems related to Bell's Theorem. Journal of Computational and Applied Mathematics. 2014. Vol. 263. P. 392–404. DOI: 10.1016/j.cam.2013.12.003.
6. Sodhi M.S. LP modeling for asset-liability management: A survey of choices and simplifications. Operations Research. 2005. Vol. 53, no. 2. P. 181–196. DOI: 10.1287/opre.1040.0185.
7. Brogaard J., Hendershott T., Riordan R. High-Frequency Trading and Price Discovery. Review of Financial Studies. 2014. Vol. 27, no. 8. P. 2267–2306. DOI: 10.1093/rfs/hhu032.
8. Dyshaev M.M., Sokolinskaya I.M. Representation of trading signals based on the Kaufman's Adaptive Moving Average in the form of a system of linear inequalities. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2014. Vol. 2, no. 4. P. 103–108. (in Russian) DOI: 10.14529/cmse130408.
9. Bixby R.E. Solving Real-World Linear Programs: A Decade and More of Progress. Operations Research. 2002. Vol. 50, no. 1. P. 3–15. DOI: 10.1287/opre.50.1.3.17780.
10. Dongarra J., Gottlieb S., Kramer W.T.C. Race to Exascale. Computing in Science and Engineering. 2019. Vol. 21, no. 1. P. 4–5. DOI: 10.1109/MCSE.2018.2882574.
11. Dantzig G.B. Linear programming and extensions. Princeton, N.J.: Princeton university press, 1998. 656 p.
12. Zadeh N. A bad network problem for the simplex method and other minimum cost flow algorithms. Mathematical Programming. 1973. Vol. 5, no. 1. P. 255–266. DOI: 10.1007/BF01580132.
13. Tolla P. A Survey of Some Linear Programming Methods. Concepts of Combinatorial Optimization / ed. by V.T. Paschos. 2nd ed. Hoboken, NJ, USA: John Wiley, Sons, 2014. Chap. 7. P. 157–188. DOI: 10.1002/9781119005216.ch7.
14. Mamalis B., Pantziou G. Advances in the Parallelization of the Simplex Method. Algorithms, Probability, Networks, and Games. Lecture Notes in Computer Science, vol. 9295 / ed. by C. Zaroliagis, G. Pantziou, S. Kontogiannis. Cham: Springer, 2015. P. 281–307. DOI: 10.1007/978-3-319-24024-4_17.
15. Karmarkar N. A new polynomial-time algorithm for linear programming. Combinatorica. 1984. Vol. 4, no. 4. P. 373–395. DOI: 10.1007/BF02579150.
16. Yuan Y. Implementation tricks of interior-point methods for large-scale linear programs. Proc. SPIE, vol. 8285. International Conference on Graphic and Image Processing (ICGIP 2011). International Society for Optics, Photonics, 2011. DOI: 10.1117/12.913019.
17. Ershova A.V., Sokolinskaya I.M. On the convergence of a scalable algorithm for constructing a pseudoprojection on a convex closed set. Bulletin of the South Ural State University, Series: Mathematical Modelling, Programming and Computer Software. 2011. No. 37(254). (in Russian).
18. Hafsteinsson H., Levkovitz R., Mitra G. Solving large scale linear programming problems using an interior point method on a massively parallel SIMD computer. Parallel Algorithms and Applications. 1994. Vol. 4, no. 3-4. P. 301–316. DOI: 10.1080/10637199408915470.
19. Karypis G., Gupta A., Kumar V. A parallel formulation of interior point algorithms. Proceedings of the 1994 ACM/IEEE conference on Supercomputing (Supercomputing'94). Los Alamitos, CA, USA: IEEE Computer Society Press, 1994. P. 204–213. DOI: 10.1109/SUPERC.1994.344280.

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

21. Schmidhuber J. Deep learning in neural networks: An overview. Neural Networks. 2015. Vol. 61. P. 85–117. DOI: 10.1016/j.neunet.2014.09.003.

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

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

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

25. Ezhova N.A., Sokolinsky L.B. Parallel computation model BSF-MR. Control systems and information technologies. 2019. No. 3(77). P. 15–21. (in Russian).

26. Bird R.S. Lectures on Constructive Functional Programming. Constructive Methods in Computing Science. NATO ASI Series F: Computer and Systems Sciences, vol. 55 / ed. by M. Broy. Berlin, Heidelberg: Springer, 1988. P. 151–216.

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

28. Sokolinsky L.B. BSF-skeleton. 2019. URL: https://github.com/leonid-sokolinsky/BSF-skeleton (accessed: 24.01.2022).

29. Gropp W. MPI 3 and Beyond: Why MPI Is Successful and What Challenges It Faces. Recent Advances in the Message Passing Interface. EuroMPI 2012. Lecture Notes in Computer Science, vol. 7490 / ed. by J. Traff, S. Benkner, J. Dongarra. Berlin, Heidelberg: Springer, 2012. P. 1–9. DOI: 10.1007/978-3-642-33518-1_1.

30. Kale V. Shared-memory Parallel Programming with OpenMP. Parallel Computing Architectures and APIs. Boca Raton: Chapman, Hall/CRC, 2019. Chap. 14. P. 213–222. DOI: 10.1201/9781351029223-18/SHARED-MEMORY-PARALLEL-PROGRAMMING-OPENMP-VIVEK-KALE.

31. Kostenetskiy P., Semenikhina P. SUSU Supercomputer Resources for Industry and fundamental Science. Proceedings - 2018 Global Smart Industry Conference, GloSIC 2018. IEEE, 2018. Article 8570068. P. 1–7. DOI: 10.1109/GloSIC.2018.8570068.

32. Sokolinsky L.B., Sokolinskaya I.M. On generation of random linear programming problems on cluster computing systems. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2021. Vol. 10, no. 1. P. 38–52. (in Russian).

33. Sokolinsky L.B. Certificate of state registration of computer program no. 2021619526 Russian Federation. Generator of random linear programming problems FRaGenLP : no. 2021618165 : application 28.05.2021 : publ. 10.06.2021. (in Russian).