ISSN 1995-0802, Lobachevskii Journal of Mathematics, 2024, Vol. 45, No. 10, pp. 5061–5079. © Pleiades Publishing, Ltd., 2024.

Surface Movement Method for Linear Programming

N. A. Olkhovsky[1]* and L. B. Sokolinsky[1]**

(Submitted by E. K. Lipachev)

[1]South Ural State University, Chelyabinsk, 454080 Russia

Received June 18, 2024; revised July 31, 2024; accepted August 13, 2024

Abstract—The article presents a new method of linear programming, called the surface movement method. This method constructs an optimal objective path on the surface of the feasible polytope from the initial boundary point to the point at which the optimal value of the objective function is achieved. The optimality of the path means moving in the direction of maximum increase/decrease in the value of the objective function. A formal description of the algorithm implementing the surface movement method is described. The convergence theorem of this algorithm is proved. The presented method can be effectively implemented using a feed forward deep neural network to determine the optimal direction of movement along the faces of the feasible polytope. To do this, a multidimensional local image of the linear programming problem is constructed at the point of the current approximation. This image is fed to the input of the deep neural network, which returns a vector determining the direction of the optimal objective path on the polytope surface.

DOI: 10.1134/S1995080224605745

Keywords and phrases: *linear programming, surface movement method, convergence theorem, deep neural network*

1. INTRODUCTION

The rapid development of big data processing technologies in the last decades [1] caused to the emergence of optimization mathematical models in the form of large-scale linear programming (LP) problems [2, 3]. Of particular interest are the LP problems related to the optimization of non-stationary processes [4]. In a non-stationary LP problem, the constraints and the objective function can change dynamically during computing its solution [5]. The following optimization problems can be reduced to non-stationary LP problems: choosing the best high-frequency trading strategies [6], optimal navigation and control of aircraft [7], dynamic optimization of batch processes [8], logistics and transportation [9–11], production planning and control [12]. Separately, we can mention optimization problems that must be solved in real-time mode [13]. Examples are the following: chemical plant control, energy management, traffic control, multi-point fuel injection system in ICE, autopilot systems, missile guidance system and others. Typically, such applications have time bounds from a few microseconds to several milliseconds.

The simplest approach to solving non-stationary optimization problems is that any changes in the input data is perceived as a separate new problem [4]. This approach may be acceptable when changes occur relatively slowly, and the optimization problem is solved relatively quickly. However, for large-scale non-stationary optimization problems, the solution obtained in this way may be far from optimal due to changes in the input data during computations. In this case, it is necessary to use algorithms that dynamically correct the computational process in accordance with the changing input data. Thus, computations with changed data do not start from scratch, but use information obtained in the past. This approach is applicable to solving optimization problems in real time, provided that the algorithm tracks

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

the movement of the optimal point quickly enough. For large-scale LP problems, the latter requirement makes it urgent to develop scalable methods and parallel algorithms for linear programming.

Up until the present time, one of the most popular ways to solve linear programming problems is a family of algorithms based on the simplex method [14]. The simplex method is capable of solving large-scale LP problems by effectively using various types of hyper-sparsity [15]. However, the simplex method has a number of fundamental drawbacks. First, in the worst case, the simplex method must visit all the vertices of the polytope that bound the feasible region, which corresponds to exponential time complexity [16–18]. Second, the use of the simplex method for solving LP problems with a dimension greater than 50 000 results in a precision loss [19] that cannot be corrected by such compute-intensive algorithms as affine scaling or iterative refinement [20]. Third, the communication structure of algorithms based on the simplex method generally has a limited degree of parallelism, which makes it impossible to efficiently parallelize them on large multiprocessor computing systems with distributed memory [21, 22]. All of the above makes it difficult to use the simplex method to solve non-stationary large-scale LP problems in real-time mode.

Another popular approach to solving large LP problems is a class of algorithms based on the interior-point method [23]. For the first time, this method was described by Dikin in [24, 25]. The interior-point method is capable of solving LP problems with millions of variables and constraints [26]. The advantage of the interior-point method is that it is self-correcting and is able to provide a high computational accuracy. The main disadvantages of the interior-point method are the following. First, an important subclass of algorithms based on the interior-point method requires some inner point of the feasible region as an initial approximation. Finding such a point can be reduced to solving an supplementary LP problem [27]. Another way to find an inner point is to use Fejér approximations [28]. Second, the interior-point method does not scale well in large cluster computing systems. There are some special cases when effective parallelization of the interior-point method is possible (see, for example, [29]), but in general it is not possible to build an efficient parallel implementation of this method for cluster computing systems. Third, the iterative nature of the interior-point method does not allow predicting the computation time in advance for a certain LP problem. These disadvantages make it difficult to use the interior-point method to solve large-scale LP problems in real-time mode.

Artificial neural networks (ANN) [30] are a promising new approach to solving optimization problems, which attracts a lot of attention. ANNs is a powerful universal tool that is applicable in almost all problem areas. One of the first applications of ANNs to solving LP problems was the work of Hopfield and Tank [31]. The Hopfield-Tank ANN consists of two fully connected layers and is recurrent. The number of neurons in the first layer is equal to the number of LP problem variables. The number of neurons in the second layer is equal to the number of constraints. The weights and biases of the ANN are completely determined by the parameters of the LP problem. The output is cyclically fed to the input of the ANN. The ANN works until an equilibrium state is reached, when the output becomes equal to the input. This equilibrium state corresponds to a minimum of a special energy function and is a solution to the LP problem. There are a lot of works that extend the Hopfield–Tank approach (see, for example, [32–36]). The main disadvantage of this approach is that it is impossible to predict the number of ANN work cycles required to achieve the equilibrium state. This makes it impossible to use such recurrent networks to solve large-scale LP problems in real-time mode. For this purpose, feed forward deep neural networks seem more promising. The structure and parameters of such networks, as a rule, do not depend on the particular input parameters of the problem. The solution is obtained in one pass with a fixed network operation time, which makes it possible to use them to solve problems in real-time mode. One of the important classes of ANNs are convolutional neural networks [37]. This class is of special interest for image processing. In recent paper [38], a new method for constructing images of multidimensional LP problems was proposed, which makes it possible to use feed forward neural networks, including convolutional ones, to solve them. It should be noted that deep neural networks require training on a large number of labeled datasets, which can be efficiently performed on GPUs [39]. In paper [40], the apex method is proposed for solving LP problems, which makes it possible to construct a path on the surface of the feasible polytope in the direction of maximum increase/decrease in the value of the objective function, leading to the optimum point. The apex method belongs to the class of iterative projection-type methods, which are characterized by a low linear convergence rate, which makes them unacceptable for real-time mode. However, the apex method allows you to construct a labeled dataset for training deep neural network.

This article presents a new method for solving LP problems, called the surface movement method. This method is intended for using feed forward ANNs, including convolutional neural networks. The rest of the paper is organized as follows. Section 2 presents the theoretical background on which the surface movement method is based. Section 3 contains a description of the surface movement method and a proof of the convergence theorem. In Section 4, we discuss the strengths and weaknesses of the proposed method, as well as reveal the ways of its practical implementation based on the synthesis of supercomputer and neural network technologies. In Conclusions, we summarize the results and provide further research directions. A summary of the main symbols used in the paper is presented in Appendix.

2. THEORETICAL BACKGROUND

This section presents a theoretical background on which the surface movement method is based. We consider a LP problem in the following form

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

where $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$, $A \in \mathbb{R}^{m\times n}$, $m > 1$, $\mathbf{c} \neq \mathbf{0}$. Here $\langle \cdot, \cdot\rangle$ stands for the dot product of two vectors. We assume that the constraint $\mathbf{x} \geqslant \mathbf{0}$ is also included in the matrix inequality $A\mathbf{x} \leqslant \mathbf{b}$ in the form of $-\mathbf{x} \leqslant \mathbf{0}$. Denote by $\mathcal{P} = \{1, \cdots, m\}$ the set of indexes numbering the rows of the matrix A. The linear objective function of the problem (1) has the form $f(\mathbf{x}) = \langle \mathbf{c}, \mathbf{x}\rangle$. In this case, the vector $\mathbf{c}$ is the gradient of the objective function $f(\mathbf{x})$.

Let $\mathbf{a}_i \in \mathbb{R}^n$ denote a vector representing the ith row of the matrix A. We assume that $\mathbf{a}_i \neq \mathbf{0}$ for all $i \in \mathcal{P}$. Denote by $\hat{H}_i$ a closed half-space defined by the inequality $\langle \mathbf{a}_i, \mathbf{x}\rangle \leqslant b_i$, and by H_i—the hyperplane bounding it:

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

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

Let us define a feasible polytope

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

representing the feasible region of LP Problem (1). Note that M, in this case, is a closed convex set. We assume that M is bounded, and $M \neq \emptyset$, i.e., LP Problem (1) has a solution. Denote by $\Gamma(M)$ the set of boundary points of the polytope M[1].

Proposition 1. *Let M be the feasible polytope of LP Problem (1), defined by equation (4). Let $V_\epsilon(\mathbf{u})$ denotes the ϵ-neighborhood of the point $\mathbf{u}$ in $\mathbb{R}^n$. Then,*

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

i.e., for any point $\mathbf{u} \in \Gamma(M)$, there exists $\epsilon > 0$ such that for each boundary point $\mathbf{w}$ belonging to the ϵ-neighborhood of the point $\mathbf{u}$, there exists at least one $i' \in \mathcal{P}$ such that $\mathbf{u}, \mathbf{w} \in H_{i'}$.

Proof. Fix an arbitrary point $\mathbf{u} \in \Gamma(M)$. Define

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

In other words, $\mathcal{P}_\mathbf{u}$ is the set of indices of all hyperplanes H_i to which the point $\mathbf{u}$ belongs. Denote $\mathcal{P}_{\backslash\mathbf{u}} = \mathcal{P}\backslash\mathcal{P}_\mathbf{u}$, i.e., $\mathcal{P}_{\backslash\mathbf{u}}$ is the set of indices of all hyperplanes H_i to which the point $\mathbf{u}$ does not belong. Define $\delta = \min\left\{ \operatorname{dist}(\mathbf{u}, H_i) |\, i \in \mathcal{P}_{\backslash\mathbf{u}}\right\}$, where $\operatorname{dist}(\mathbf{u}, H_i)$ stands for the Euclidean distance from the point $\mathbf{u}$ to the hyperplane H_i[2]. By definition, $\delta > 0$. Take ϵ satisfying the condition $0 < \epsilon < \delta$. Then, for any $\mathbf{w} \in V_\epsilon(\mathbf{u}) \cap \Gamma(M)$, the following condition holds

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

[1] If $M \subset \mathbb{R}^n$, then a point $\mathbf{x} \in \mathbb{R}^n$ is a boundary point of M if every neighborhood of $\mathbf{x}$ contains at least one point in M and at least one point not in M.

[2] In this case, $\operatorname{dist}(\mathbf{u}, H_i) = \frac{\langle \mathbf{a}_i, \mathbf{u}\rangle - b_i}{||\mathbf{a}_i||}$.

Since $\mathbf{w}$ is a boundary point, it follows that there exists an $i' \in \mathcal{P}_\mathbf{u}$ such that $\mathbf{w} \in H_{i'}$. By virtue of (5), the following condition also holds $\mathbf{u} \in H_{i'}$. □

Definition 1. The objective projection of a point $\mathbf{z} \in \mathbb{R}^n$ onto the hyperplane H_i is a point $\boldsymbol{\gamma}_i(\mathbf{z}) \in \mathbb{R}^n \cup \{\infty\}$ defined by the equation

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

where $L(\mathbf{z})$ is the line passing through the point $\mathbf{z}$ parallel to the vector $\mathbf{c}$:

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

In other words, if the vector $\mathbf{c}$ is not parallel to the hyperplane H_i, then the objective projection of the point $\mathbf{z}$ onto the hyperplane H_i is the intersection point of this hyperplane with the line passing through the point $\mathbf{z}$ parallel to the vector $\mathbf{c}$. In the case when the vector $\mathbf{c}$ is parallel to the hyperplane H_i, the objective projection is the point at infinity.

The following proposition provides an equation for calculating the objective projection $\boldsymbol{\gamma}_i(\mathbf{z})$ of point $\mathbf{z}$ onto hyperplane H_i.

Proposition 2. *Let $\langle \mathbf{a}_i, \mathbf{c}\rangle \neq 0$, i.e., the vector $\mathbf{c}$ is not parallel to the hyperplane H_i. Then,*

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

Proof. In accordance with (6) and (7), the following equation holds for some $\lambda \in \mathbb{R}$:

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

On the other hand, in accordance with (3), the following equation holds

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

Substituting the right-hand side of equation (9) instead of $\boldsymbol{\gamma}_i(z)$ into equation (10), we obtain $\langle \mathbf{a}_i, \mathbf{z} + \lambda \mathbf{c}\rangle = b_i$. It follows that

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

Substituting the right-hand side of equation (11) instead of λ into equation (9), we obtain

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

□

Definition 2. The objective bias of the point $\mathbf{z} \in \mathbb{R}^n$ relative to the hyperplane H_i is the scalar quantity $\beta_i(\mathbf{z})$, calculated by the equation

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

For brevity's sake, everywhere below, we will use the term "bias", meaning by this "objective bias". Denote

$$\mathbf{e_c} = \mathbf{c}/\|\mathbf{c}\|. \tag{13}$$

Then, equation (8) can be rewritten as follows:

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

which is equivalent to $\beta_i(\mathbf{z})\mathbf{e_c} = \boldsymbol{\gamma}_i(\mathbf{z}) - \mathbf{z}$. Taking into account (13), it follows

$$|\beta_i(\mathbf{z})| = \|\boldsymbol{\gamma}_i(\mathbf{z}) - \mathbf{z}\|. \tag{15}$$

Thus, $|\beta_i(\mathbf{z})|$ is the distance from the point $\mathbf{z}$ to its objective projection onto the hyperplane H_i.

Definition 3. The objective hyperplane $H_c(\mathbf{z})$ passing through the point $\mathbf{z}$ is the hyperplane defined by the following equation

$$H_c(\mathbf{z}) = \{\mathbf{x} \in \mathbb{R}^n | \langle \mathbf{c}, \mathbf{x} \rangle = \langle \mathbf{c}, \mathbf{z} \rangle \}. \tag{16}$$

The following proposition holds.

Proposition 3. *Fix an arbitrary point $\mathbf{z} \in \mathbb{R}^n$. Then, for any points $\mathbf{z}', \mathbf{z}'' \in H_c(\mathbf{z})$, $\mathbf{z}' \neq \mathbf{z}''$, the following statement is true for all $i \in \mathcal{P}$:*

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

Proof. Since $\mathbf{z}', \mathbf{z}'' \in H_c(\mathbf{z})$, and $\mathbf{c}$ is normal to the hyperplane $H_c(\mathbf{z})$, the following two equations hold $\langle \mathbf{c}, \mathbf{z}' - \mathbf{z} \rangle = 0$;

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

$$\left\langle \mathbf{c}, \mathbf{z}'' \right\rangle = \langle \mathbf{c}, \mathbf{z} \rangle = \left\langle \mathbf{c}, \mathbf{z}' \right\rangle. \tag{17}$$

Sequentially using (14), (17), and (13), we obtain the following chain of equivalent inequalities:

$$\left\langle \mathbf{c}, \boldsymbol{\gamma}_i(\mathbf{z}') \right\rangle < \left\langle \mathbf{c}, \boldsymbol{\gamma}_i(\mathbf{z}'') \right\rangle \Leftrightarrow \left\langle \mathbf{c}, \mathbf{z}' + \beta_i(\mathbf{z}')\mathbf{e}_\mathbf{c} \right\rangle < \left\langle \mathbf{c}, \mathbf{z}'' + \beta_i(\mathbf{z}'')\mathbf{e}_\mathbf{c} \right\rangle$$
$$\Leftrightarrow \left\langle \mathbf{c}, \beta_i(\mathbf{z}')\mathbf{e}_\mathbf{c} \right\rangle < \left\langle \mathbf{c}, \beta_i(\mathbf{z}'')\mathbf{e}_\mathbf{c} \right\rangle \Leftrightarrow \left\langle \mathbf{c}, \beta_i(\mathbf{z}')\mathbf{c}/\left|\left|\mathbf{c}\right|\right| \right\rangle < \left\langle \mathbf{c}, \beta_i(\mathbf{z}'')\mathbf{c}/\left|\left|\mathbf{c}\right|\right| \right\rangle$$
$$\Leftrightarrow \frac{\beta_i(\mathbf{z}')}{||\mathbf{c}||} \langle \mathbf{c}, \mathbf{c} \rangle < \frac{\beta_i(\mathbf{z}'')}{||\mathbf{c}||} \langle \mathbf{c}, \mathbf{c} \rangle \Leftrightarrow \beta_i(\mathbf{z}') < \beta_i(\mathbf{z}'').$$

□

The following definition of a recessive half-space is adopted by us from the paper [40].

Definition 4. The half-space $\hat{H}_i$ is called recessive, if

$$\forall \mathbf{x} \in H_i, \quad \forall \lambda > 0 : \mathbf{x} + \lambda \mathbf{c} \notin \hat{H}_i. \tag{18}$$

The geometric interpretation of this definition is as follows: the ray parallel to the vector $\mathbf{c}$ coming from any point of the hyperplane bounding the recessive half-space has no points in common with this half-space, except for the starting one.

The following condition is both necessary and sufficient for the half-space $\hat{H}_i$ to be recessive [40]:

$$\langle \mathbf{a}_i, \mathbf{c} \rangle > 0. \tag{19}$$

A recessive half-space has the following properties.

Property 1. *Let the half-space $\hat{H}_i$ be recessive. Then, any line parallel to the vector $\mathbf{c}$ intersects the hyperplane H_i at a single point.*

This property directly follows from the fact that the hyperplane H_i bounding the recessive half-space $\hat{H}_i$ cannot be parallel to the vector $\mathbf{c}$ by Definition 4.

Property 2. *Let the half-space $\hat{H}_i$ be recessive. Then,*

$$\mathbf{x} \in \hat{H}_i \Leftrightarrow \beta_i(\mathbf{x}) \geqslant 0. \tag{20}$$

Proof. First, assume that $\mathbf{x} \in \hat{H}_i$. Then, according to (2), the following inequality holds $\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i \leqslant 0$. By virtue of (12), we have

$$\beta_i(\mathbf{x}) = -\frac{\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i}{\langle \mathbf{a}_i, \mathbf{c} \rangle} \left|\left|\mathbf{c}\right|\right|. \tag{21}$$

Taking into account (19), we obtain $\mathbf{x} \in \hat{H}_i \Rightarrow \beta_i(\mathbf{x}) \geqslant 0$.

Now, suppose that $\beta_i(\mathbf{x}) \geqslant 0$. According to (12), this means that

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

Given (19), we obtain from here $\langle \mathbf{a}_i, \mathbf{x} \rangle - b_i \leqslant 0$. According to (2), it follows that $\mathbf{x} \in \hat{H}_i$. Thus,

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

□

Define

$$\mathcal{I} = \{i \in \mathcal{P} \,|\langle \mathbf{a}_i, \mathbf{c}\rangle > 0\}, \tag{22}$$

i.e., $\mathcal{I}$ represents a set of indices for which the half-space $\hat{H}_i$ is recessive. Since the feasible polytope M is a bounded set, we have

$$\mathcal{I} \neq \emptyset. \tag{23}$$

Denote

$$\hat{M} = \bigcap_{i\in\mathcal{I}} \hat{H}_i. \tag{24}$$

Obviously, $\hat{M}$ is a convex, closed, unbounded polytope. Let us call it a recessive polytope. By (4) and (22), it follows that

$$M \subset \hat{M}. \tag{25}$$

Let $\Gamma(\hat{M})$ denote the set of boundary points of the recessive polytope $\hat{M}$. According to Proposition 3 in [40], we have $\bar{\mathbf{x}} \in \Gamma(\hat{M})$, i.e., the solution of LP Problem (1) lies on the boundary of the recessive polytope $\hat{M}$.

Proposition 4. *Let $\hat{M}$ be the recessive polytope defined by equation (24). Then, for any point $\mathbf{u} \in \Gamma(\hat{M})$, there exists $\epsilon > 0$ such that for any boundary point $\mathbf{w}$ belonging to the ϵ-neighborhood $V_\epsilon(\mathbf{u})$ of the point $\mathbf{u}$, there is $i' \in \mathcal{I}$ for which $\mathbf{u}, \mathbf{w} \in H_{i'}$ is valid, i.e.,*

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

Proof. The proof is similar to the proof of Proposition 1. □

Definition 5. The objective projection of the point $\mathbf{z} \in \mathbb{R}^n$ onto the boundary $\Gamma(\hat{M})$ of the recessive polytope $\hat{M}$ is the point $\hat{\boldsymbol{\gamma}}(\mathbf{z})$ calculated by the equation $\hat{\boldsymbol{\gamma}}(\mathbf{z}) = L(\mathbf{z}) \cap \Gamma(\hat{M})$, where $L(\mathbf{z})$ is the line passing through the point $\mathbf{z}$ parallel to the vector $\mathbf{c}$:

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

The scalar quantity $\hat{\beta}(\mathbf{z}) \in \mathbb{R}$ satisfying the equation

$$\hat{\boldsymbol{\gamma}}(\mathbf{z}) = \mathbf{z} + \hat{\beta}(\mathbf{z})\mathbf{c} \tag{26}$$

will be called the bias of the point $\mathbf{z}$ relative to the boundary of the recessive polytope $\hat{M}$.

Note that the correctness of this definition is based on property 1. The following proposition provides an equation for calculating the objective projection onto the boundary of the recessive polytope. Figure 1 illustrates the proof of this proposition.

Proposition 5. *Let an arbitrary point $\mathbf{z} \in \mathbb{R}^n$ be given. Put*

$$i' = \arg\min \{\beta_i(\mathbf{z}) \,| i \in \mathcal{I}\}. \tag{27}$$

Then,

$$\hat{\boldsymbol{\gamma}}(\mathbf{z}) = \boldsymbol{\gamma}_{i'}(\mathbf{z}). \tag{28}$$

In other words, the objective projection of the point $\mathbf{z}$ onto the boundary of the recessive polytope $\hat{M}$ coincides with the projection of this point onto the hyperplane $H_{i'}$, which has the minimum bias relative to $\mathbf{z}$.

Proof. Fix an arbitrary point $\mathbf{z} \in \mathbb{R}^n$. According to Definition 5, let us construct a line parallel to the vector $\mathbf{c}$, which passes through the point $\mathbf{z}$: $L = \{\mathbf{y} \in \mathbb{R}^n |\, \mathbf{y} = \mathbf{z} + \lambda\mathbf{c}, \lambda \in \mathbb{R}\}$. The following equation holds: $\hat{\boldsymbol{\gamma}}(\mathbf{z}) = L \cap \Gamma(\hat{M})$. In accordance with property 1, for any $i \in \mathcal{I}$, the line L intersects the hyperplane H_i at only one point, which we denote as $\mathbf{y}_i$: $L \cap H_i = \{\mathbf{y}_i\}$. Define $Y = \bigcup\limits_{i\in\mathcal{I}} \{\mathbf{y}_i\}$, i.e., Y

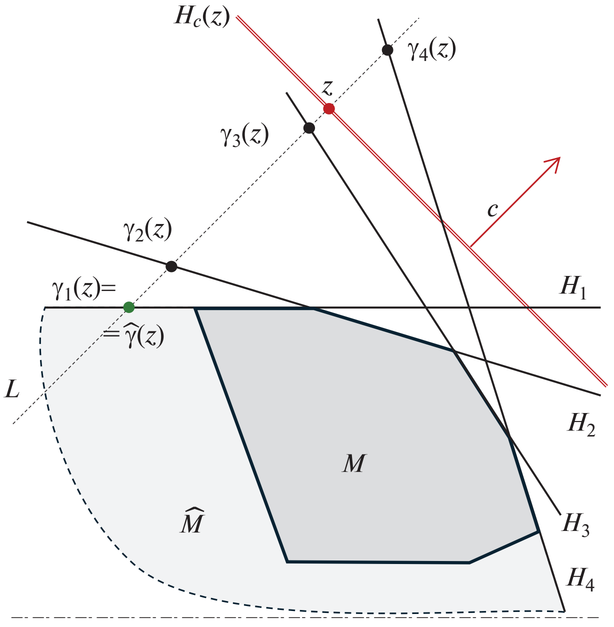

Fig. 1. Illustration to proof of Proposition 5.

is the set of points at which the line L intersects the boundaries of recessive half-spaces. According to Definition 1, the following conditions hold

$$\forall i \in \mathcal{I} \,:\, \boldsymbol{\gamma}_i(\mathbf{z}) = \mathbf{y}_i, \tag{29}$$

and

$$\forall i \in \mathcal{I} \,:\, \mathbf{y}_i \in H_i. \tag{30}$$

By virtue of (24), the following is also true $\hat{\boldsymbol{\gamma}}(\mathbf{z}) \in Y$. This means that there is $i' \in \mathcal{I}$ such that

$$\hat{\boldsymbol{\gamma}}(\mathbf{z}) = \boldsymbol{\gamma}_{i'}(\mathbf{z}). \tag{31}$$

Let us show that $i' \in \arg\min\{\beta_i(\mathbf{z}) | i \in \mathcal{I}\}$. Assume the opposite, namely that $\beta_{i'}(\mathbf{z}) > \min\{\beta_i(\mathbf{z}) | i \in \mathcal{I}\}$. Then, there exists an $i'' \in \mathcal{I}$ such that

$$\beta_{i''}(\mathbf{z}) < \beta_{i'}(\mathbf{z}). \tag{32}$$

By (14) and (29),

$$\mathbf{y}_{i'} = \mathbf{z} + \beta_{i'}(\mathbf{z})\mathbf{e_c}; \quad \mathbf{y}_{i''} = \mathbf{z} + \beta_{i''}(\mathbf{z})\mathbf{e_c}.$$

Hence,

$$\mathbf{y}_{i'} = \mathbf{y}_{i''} + \left(\beta_{i'}(\mathbf{z}) - \beta_{i''}(\mathbf{z})\right)\mathbf{e_c},$$

which, by virtue of (32) and (13) is equivalent to

$$\mathbf{y}_{i'} = \mathbf{y}_{i''} + \frac{|\beta_{i''}(\mathbf{z}) - \beta_{i'}(\mathbf{z})|}{||\mathbf{c}||}\mathbf{c}. \tag{33}$$

In accordance with (18) and (30), it follows from (33) that $\mathbf{y}_{i'} \notin \hat{H}_{i''}$. This means that $\mathbf{y}_{i'} \notin \hat{M}$. Taking into account (29) and (31), it follows that $\hat{\boldsymbol{\gamma}}(\mathbf{z}) \notin \hat{M}$. We have thus reached a contradiction to Definition 5. □

The following proposition provides an equation for calculating the bias of a point relative to the boundary of a recessive polytope.

Proposition 6. *Let an arbitrary point* $\mathbf{z} \in \mathbb{R}^n$ *be given. Then,*

$$\hat{\beta}(\mathbf{z}) = \frac{\langle \mathbf{c}, \hat{\boldsymbol{\gamma}}(\mathbf{z}) - \mathbf{z} \rangle}{||\mathbf{c}||^2}. \tag{34}$$

Algorithm 1 Surface movement method

```
Require: $\forall \mathbf{z} \in \mathbb{R}^n : H_c(\mathbf{z}) = \{\mathbf{x} \in \mathbb{R}^n | \langle \mathbf{c}, \mathbf{x} - \mathbf{z}\rangle = 0\}$; $\hat{H}_i = \{\mathbf{x} \in \mathbb{R}^n | \langle \mathbf{a}_i, \mathbf{x}\rangle \leqslant b_i\}$; $\hat{M} = \bigcap_{i \in \mathcal{I}} \hat{H}_i$
 1: input $\mathbf{u}^{(0)}$
 2: assert $\mathbf{u}^{(0)} \in M \cap \Gamma(\hat{M})$
 3: $k := 0$
 4: $r := 0.1$
 5: $D^{(0)} := H_c(\mathbf{u}^{(0)}) \cap V_r(\mathbf{u}^{(0)})$        ▷ $V_r(\mathbf{u}^{(0)})$ is $r$-neighborhood of point $\mathbf{u}^{(0)}$
 6: $\mathbf{v}^{(0)} := \arg\max\{\hat{\beta}(\mathbf{z}) \mid \mathbf{z} \in D^{(0)}\}$
 7: $\mathbf{w}^{(0)} := \hat{\boldsymbol{\gamma}}(\mathbf{v}^{(0)})$
 8: assert $\exists i \in \mathcal{I} : \mathbf{w}^{(0)}, \mathbf{u}^{(0)} \in H_i \cap \Gamma(\hat{M})$        ▷ If this fails, reduce $r$
 9: while $\langle \mathbf{c}, \mathbf{w}^{(k)} - \mathbf{u}^{(k)}\rangle > \epsilon_f$
10:     assert $\exists i \in \mathcal{I} : \mathbf{w}^{(k)}, \mathbf{u}^{(k)} \in H_i \cap \Gamma(\hat{M})$        ▷ If this fails, reduce $r$
11:     $\mathbf{d}^{(k)} := \mathbf{w}^{(k)} - \mathbf{u}^{(k)}$
12:     $L^{(k)} := \{\mathbf{u}^{(k)} + \lambda \mathbf{d}^{(k)} \mid \lambda \in \mathbb{R}_{>0}\}$
13:     $\mathbf{u}^{(k+1)} := \arg\max\{||\mathbf{x} - \mathbf{u}^{(k)}|| \mid \mathbf{x} \in L^{(k)} \cap \Gamma(M)\}$
14:     $D^{(k+1)} := H_c(\mathbf{u}^{(k+1)}) \cap V_r(\mathbf{u}^{(k+1)})$
15:     $\mathbf{v}^{(k+1)} := \arg\max\{\hat{\beta}(\mathbf{z}) \mid \mathbf{z} \in D^{(k+1)}\}$
16:     $\mathbf{w}^{(k+1)} := \hat{\boldsymbol{\gamma}}(\mathbf{v}^{(k+1)})$
17:     $k := k + 1$
18: end while
19: output $\mathbf{u}^{(k)}$
20: stop
```

Proof. In accordance with (26), the points $\hat{\boldsymbol{\gamma}}(\mathbf{z})$ and $\mathbf{z}$ are on the normal to the hyperplane $H_c(\mathbf{z})$. Therefore, the point $\mathbf{z} \in H_c(\mathbf{z})$ is an orthogonal projection of the point $\hat{\boldsymbol{\gamma}}(\mathbf{z})$ onto the hyperplane $H_c(\mathbf{z})$, and taking into account (16), it can be calculated by the following known equation

$$\mathbf{z} = \hat{\boldsymbol{\gamma}}(\mathbf{z}) - \frac{\langle \mathbf{c}, \hat{\boldsymbol{\gamma}}(\mathbf{z}) - \mathbf{z}\rangle}{||\mathbf{c}||^2}\mathbf{c}. \tag{35}$$

We can rewrite this as $\hat{\boldsymbol{\gamma}}(\mathbf{z}) = \mathbf{z} + \frac{\langle \mathbf{c}, \hat{\boldsymbol{\gamma}}(\mathbf{z}) - \mathbf{z}\rangle}{||\mathbf{c}||^2}\mathbf{c}$. Comparing this with (26), we obtain

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

□

3. SURFACE MOVEMENT METHOD

The surface movement method constructs, on the surface of the feasible polytope, a path from an arbitrary boundary point $\mathbf{u}^{(0)} \in M \cap \Gamma(\hat{M})$ to a point $\bar{\mathbf{x}}$, which is a solution to LP Problem (1). Moving along the surface of the recessive polytope is performed in the direction of the greatest increase in the value of the objective function. The path constructed as a result of such a movement will be called the optimal objective path. The implementation of the surface movement method is presented in the form of Algorithm 1.

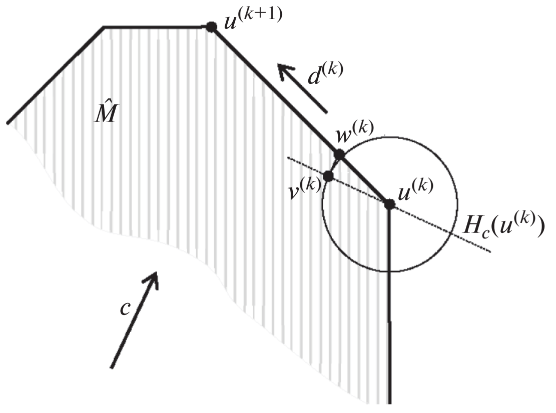

Fig. 2. Surface movement method.

Let us give brief comments on this implementation. Step 1 reads the initial approximation of $\mathbf{u}^{(0)}$. It can be an arbitrary boundary point of a recessive polytope $\hat{M}$ satisfying the condition

$$\mathbf{u}^{(0)} \in M \cap \Gamma(\hat{M}),$$

which is checked in Step 2. Step 3 assigns the iteration counter k the value 0. Step 4 sets the value of the parameter r. Step 5 builds an n-dimensional disk $D^{(0)}$, which is the intersection of the objective hyperplane $H_c\left(\mathbf{u}^{(0)}\right)$ passing through the point $\mathbf{u}^{(0)}$, and the n-dimensional ball $V_r\left(\mathbf{u}^{(0)}\right)$ of a small radius r with the center at the point $\mathbf{u}^{(0)}$. Step 6 calculates the point $\mathbf{v}^{(0)} \in D^{(0)}$, having the maximum bias relative to the boundary of the recessive polytope $\hat{M}$. The bias $\hat{\beta}(\mathbf{z})$ is calculated by equation (34). The objective projection $\hat{\gamma}(\mathbf{z})$, in equation (34), is calculated using equations (27), (28), and (8). Step 7 calculates the point $\mathbf{w}^{(0)}$, which is the objective projection of the point $\mathbf{v}^{(0)}$ on the boundary of the recessive polytope. Step 8 checks that there exists a recessive half-space $\hat{H}_i$ such that the boundary points $\mathbf{w}^{(0)}$ and $\mathbf{u}^{(0)}$ lie on the hyperplane H_i bounding this half-space. This is necessary so that the movement is carried out on the surface of the recessive polytope, and not through its interior. If this requirement is not met, it is necessary to reduce the radius r of the n-dimensional ball $V_r\left(\mathbf{u}^{(0)}\right)$. A suitable r exists by virtue of Proposition 4. Steps 9–18 implement the main loop of the surface movement method, the geometric interpretation of which is shown in Fig. 2. This loop is executed while the following condition is true:

$$\left\langle \mathbf{c}, \mathbf{w}^{(k)} - \mathbf{u}^{(k)} \right\rangle > \epsilon_f, \tag{36}$$

where ϵ_f is a small positive parameter. Step 10 checks that there exists a recessive half-space $\hat{H}_i$ such that the boundary points $\mathbf{w}^{(k)}$ and $\mathbf{u}^{(k)}$ lie on the hyperplane H_i bounding this half-space. If this requirement is not met, it is necessary to reduce the radius r of the n-dimensional ball $V_r\left(\mathbf{u}^{(k)}\right)$. Step 11 calculates the vector $\mathbf{d}^{(k)}$, which determines the movement direction. Step 12 builds the ray $L^{(k)}$ parallel to the vector $\mathbf{d}^{(k)}$ with the initial point $\mathbf{u}^{(k)}$. Step 13 determines the next approximation $\mathbf{u}^{(k+1)}$ as the point belonging to the ray $L^{(k)}$, which is lying on the boundary of the recessive polytope M as far as possible from the point $\mathbf{u}^{(k)}$. Step 14 builds the hyperdisk $D^{(k+1)}$ of radius r with the center at the point $\mathbf{u}^{(k+1)}$, lying on the hyperplane $H_c\left(\mathbf{u}^{(k+1)}\right)$. Step 15 finds the point $\mathbf{v}^{(k+1)}$ on the hyperdisk $D^{(k+1)}$ that has the maximum bias. Step 16 calculates the point $\mathbf{w}^{(k+1)}$, which is the objective projection of the point $\mathbf{v}^{(k+1)}$ onto the boundary of the recessive polytope $\hat{M}$. In Step 17, the iteration counter k is incremented by 1. Step 18 passes the control to the beginning of the `while` loop. Step 19 outputs the last approximation $\mathbf{u}^{(k)}$ as a result. Step 20 terminates the algorithm.

Note that by construction of Algorithm 1, for any k the following condition holds

$$\mathbf{u}^{(k)} \in M \cap \Gamma(\hat{M}), \tag{37}$$

i.e., all points of the sequence $\{\mathbf{u}^{(k)}\}$ generated by Algorithm 1 simultaneously lie on the boundary of the feasible polytope M and on the boundary of the recessive polytope $\hat{M}$. Also, it follows from (36) that

$$\left\langle \mathbf{c}, \mathbf{u}^{(k)} \right\rangle < \left\langle \mathbf{c}, \mathbf{w}^{(k)} \right\rangle. \tag{38}$$

In addition, the following inequality holds

$$\left\langle \mathbf{c}, \mathbf{w}^{(k)} \right\rangle \leqslant \left\langle \mathbf{c}, \mathbf{u}^{(k+1)} \right\rangle. \tag{39}$$

The following lemma guarantees that Algorithm 1 stops in a finite number of iterations.

Lemma 1. *Let the feasible polytope M of the problem (1) be a bounded nonempty set. Then, the sequence of points $\{\mathbf{u}^{(k)}\}$ generated by Algorithm 1 is finite for any $\epsilon_f > 0$.*

Proof. Assume the opposite, that is, Algorithm 1 generates the infinite sequence of points $\{\mathbf{u}^{(k)}\}$. But then, by virtue of (38) and (39), we obtain the infinite strictly monotonically increasing numerical sequence $\{\langle \mathbf{c}, \mathbf{u}^{(k)} \rangle\}_{k=0}^{\infty}$:

$$\left\langle \mathbf{c}, \mathbf{u}^{(k)} \right\rangle < \left\langle \mathbf{c}, \mathbf{u}^{(k+1)} \right\rangle. \tag{40}$$

Since, by the lemma condition, the feasible polytope M is a nonempty bounded set, there is a solution $\bar{\mathbf{x}}$ to the problem LP (1). By virtue of (37), the following inequality holds $\langle \mathbf{c}, \mathbf{u}^{(k)} \rangle \leqslant \langle \mathbf{c}, \bar{\mathbf{x}} \rangle$ for all $k = 0, 1, 2, \ldots$ This means that the sequence $\{\langle \mathbf{c}, \mathbf{u}^{(k)} \rangle\}_{k=0}^{\infty}$ is bounded from above. According to the monotone convergence theorem, a monotonically increasing bounded from above sequence converges to its supremum, i.e., there exists $k' \in \mathbb{N}$ such that

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

By (39) it follows that

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

This is equivalent to

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

We obtain a contradiction with the condition of the `while` loop (Step 9 of Algorithm 1). □

The following theorem shows that, for a sufficiently small ϵ_f, the sequence of points $\{\mathbf{u}^{(k)}\}$ calculated by Algorithm 1 converges to the solution $\bar{\mathbf{x}}$ of LP Problem (1).

Theorem 7 (Convergence of Algorithm 1). *Let the feasible polytope M of the problem (1) be a bounded nonempty set. Let $\bar{\mathbf{x}}$ be a solution of LP Problem (1). Let $\{\epsilon_\eta\}_{\eta=1}^{\infty}$ be a monotonically decreasing sequence of positive numbers converging to zero:*

$$\lim_{\eta \to \infty} \epsilon_\eta = 0. \tag{41}$$

Denote by $u^{(K_\eta)}$ the final point generated by Algorithm 1 with $\epsilon_f = \epsilon_\eta$ (it exists by virtue of Lemma 1). Then, there exists $\bar{\eta} \in \mathbb{N}$ such that, for all $\eta \geqslant \bar{\eta}$, the following equation holds

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

Proof. Let us show that the sequence $\{\langle \mathbf{c}, \mathbf{u}^{(K_\eta)} \rangle\}_{\eta=1}^{\infty}$ is monotonically increasing. Indeed, it follows from (38) and (39) that

$$\forall k' \leqslant k'' : \left\langle \mathbf{c}, \mathbf{u}^{(k')} \right\rangle \leqslant \left\langle \mathbf{c}, \mathbf{u}^{(k'')} \right\rangle. \tag{42}$$

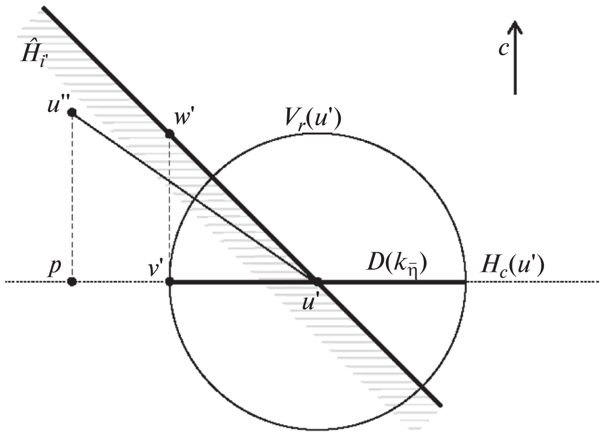

Fig. 3. Illustration to proof of Theorem 7.

By the condition of the theorem, $\epsilon_\eta \geqslant \epsilon_{\eta+1}$. Therefore, by construction of Algorithm 1, $K_\eta \leqslant K_{\eta+1}$. Comparing this with (42), we obtain

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

i.e., the sequence $\left\{\left\langle \mathbf{c}, \mathbf{u}^{(K_\eta)} \right\rangle\right\}_{\eta=1}^{\infty}$ is monotonically increasing. Obviously, this sequence is bounded from above by the value of $\langle \mathbf{c}, \bar{\mathbf{x}} \rangle$. Hence, it has a finite limit $\lim\limits_{\eta\to\infty} \left\langle \mathbf{c}, \mathbf{u}^{(K_\eta)} \right\rangle = \bar{f}$. Algorithm 1, within each iteration, passes[3] one face/edge of the recessive polytope $\hat{M}$ in the direction of maximizing the value of the objective function. In this case, each face/edge is traversed no more than once, since the polytope $\hat{M}$ is a convex set. This means that there exists $\bar{\eta} \in \mathbb{N}$ such that for all $\eta \geqslant \bar{\eta}$ the following equation holds: $\mathbf{u}^{(K_\eta)} = \mathbf{u}^{(K_{\bar{\eta}})}$, and $\left\langle \mathbf{c}, \mathbf{u}^{(K_\eta)} \right\rangle = \bar{f}$. By construction of Algorithm 1, taking into account (41), this is possible only when

$$\left\langle \mathbf{c}, \mathbf{w}^{(K_{\bar{\eta}})} - \mathbf{u}^{(K_{\bar{\eta}})} \right\rangle = 0. \tag{43}$$

Let us show that, in this case, $\left\langle \mathbf{c}, \mathbf{u}^{(K_{\bar{\eta}})} \right\rangle = \langle \mathbf{c}, \bar{\mathbf{x}} \rangle$, i.e., the point $\mathbf{u}^{(K_{\bar{\eta}})}$ is a solution of LP Problem (1). Denote $\mathbf{u}' = \mathbf{u}^{(K_{\bar{\eta}})}$, and assume the opposite, namely that there exists a point

$$\mathbf{u}'' \in M \tag{44}$$

such that $\langle \mathbf{c}, \mathbf{u}'' \rangle > \langle c, \mathbf{u}' \rangle$. This is equivalent to

$$\left\langle \mathbf{c}, \mathbf{u}'' - \mathbf{u}' \right\rangle > 0. \tag{45}$$

Figure 3 illustrates the following part of the proof.

Based on Definition 3, we can calculate the orthogonal projection $\mathbf{p}$ of the point $\mathbf{u}''$ onto the objective hyperplane $H_c(\mathbf{u}')$ passing through the point $\mathbf{u}'$:

$$\mathbf{p} = \mathbf{u}'' - \frac{\langle \mathbf{c}, \mathbf{u}'' - \mathbf{u}' \rangle}{||\mathbf{c}||^2}\mathbf{c}. \tag{46}$$

Note that

$$\left|\left|\mathbf{p} - \mathbf{u}'\right|\right| \neq 0, \tag{47}$$

since otherwise, according to Definition 4, the point $\mathbf{u}''$ cannot belong to the recessive polytope $\hat{M}$, which contradicts (44). Choose $r \in \mathbb{R}$ satisfying the condition

$$r > 0, \tag{48}$$

[3]The passage of a polytope face/edge is understood as movement inside a linear manifold of dimension k in the presence of k degrees of freedom.

for which there is $i' \in \mathcal{I}$ such that

$$\hat{\boldsymbol{\gamma}}(\mathbf{v}'), \mathbf{u}' \in H_{i'} \cap \Gamma(M), \tag{49}$$

where

$$\mathbf{v}' = \mathbf{u}' + \frac{r}{||\mathbf{p} - \mathbf{u}'||}(\mathbf{p} - \mathbf{u}'). \tag{50}$$

This is possible by virtue of Proposition 1. Without loss of generality, we can assume that r satisfies all asserts in the case when $\epsilon_f = \epsilon_{\bar{\eta}}$. Thus,

$$\mathbf{v}' \in D^{(K_{\bar{\eta}})}, \tag{51}$$

where $D^{K_{\bar{\eta}}}$ is the disk constructed in Step 14 of Algorithm 1 with $k = K_{\bar{\eta}} - 1$. Note that

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

since otherwise $\hat{\boldsymbol{\gamma}}(\mathbf{v}') \notin H_{i'}$, which contradicts (49). According to Proposition 5, it follows that

$$\hat{\boldsymbol{\gamma}}(\mathbf{v}') = \boldsymbol{\gamma}_{i'}(\mathbf{v}'). \tag{52}$$

Set $\mathbf{w}' = \hat{\boldsymbol{\gamma}}(\mathbf{v}')$. Taking into account (52), the following equation holds $\mathbf{w}' = \boldsymbol{\gamma}_{i'}(\mathbf{v}')$. Using Proposition 2, we obtain

$$\mathbf{w}' = \mathbf{v}' - \frac{\langle \mathbf{a}_{i'}, \mathbf{v}' \rangle - b_{i'}}{\langle \mathbf{a}_{i'}, \mathbf{c} \rangle}\mathbf{c}. \tag{53}$$

Since $\mathbf{u}' \in H_{i'}$, it follows from (3) that

$$\left\langle \mathbf{a}_{i'}, \mathbf{u}' \right\rangle = b_{i'}. \tag{54}$$

Therefore, (53) can be rewritten in the form

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

Substituting the right side of equation (50) instead of $\mathbf{v}'$, we obtain from here

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

which is equivalent to

$$\mathbf{w}' = \mathbf{u}' + \frac{r}{||\mathbf{p} - \mathbf{u}'||}(\mathbf{p} - \mathbf{u}') - \frac{\left\langle \mathbf{a}_{i'}, \frac{r}{||\mathbf{p} - \mathbf{u}'||}(\mathbf{p} - \mathbf{u}') \right\rangle}{\langle \mathbf{a}_{i'}, \mathbf{c} \rangle}\mathbf{c}. \tag{55}$$

According to (2), we have

$$\hat{H}_{i'} = \{\mathbf{x} \in \mathbb{R}^n | \langle \mathbf{a}_{i'}, \mathbf{x} \rangle \leqslant b_{i'} \}. \tag{56}$$

Using (54), equation (56) can be rewritten in the form

$$\hat{H}_{i'} = \left\{ \mathbf{x} \in \mathbb{R}^n \middle| \langle \mathbf{a}_{i'}, \mathbf{x} \rangle \leqslant \left\langle \mathbf{a}_{i'}, \mathbf{u}' \right\rangle \right\}. \tag{57}$$

From (44), it follows that $\mathbf{u}'' \in \hat{H}_{i'}$. Comparing this with (57), we obtain $\langle \mathbf{a}_{i'}, \mathbf{u}'' \rangle \leqslant \langle \mathbf{a}_{i'}, \mathbf{u}' \rangle$, which is equivalent to

$$\left\langle \mathbf{a}_{i'}, \mathbf{u}' - \mathbf{u}'' \right\rangle \geqslant 0. \tag{58}$$

Since the half-space $\hat{H}_{i'}$ is recessive, according to Proposition 1 in [40], the following inequality holds

$$\langle \mathbf{a}_{i'}, \mathbf{c} \rangle > 0. \tag{59}$$

By virtue of (55) and (46) we have

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

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

According to (48), (47), (59), (58), and (45), it follows that $\langle \mathbf{c},\mathbf{w}'-\mathbf{u}'\rangle > 0$. Recalling that $\mathbf{u}'=\mathbf{u}^{(K_{\bar{\eta}})}$, we rewrite the last inequality in the form

$$\left\langle \mathbf{c},\mathbf{w}'-\mathbf{u}^{(K_{\bar{\eta}})}\right\rangle > 0. \tag{60}$$

Taking in account Proposition 3, by construction of Algorithm 1 (steps 15 and 16), we have $\left\langle \mathbf{c},\mathbf{w}^{(K_{\bar{\eta}})}\right\rangle \geqslant \langle \mathbf{c},\mathbf{w}'\rangle$. Comparing this with (60), we obtain $\left\langle \mathbf{c},\mathbf{w}^{(K_{\bar{\eta}})}-\mathbf{u}^{(K_{\bar{\eta}})}\right\rangle > 0$. We have thus reached a contradiction with (43). □

4. DISCUSSION

This section discusses the strengths and weaknesses of the proposed method, as well as reveals the ways of its practical implementation based on the synthesis of high-performance computing and neural network technologies.

First of all, let us look at the issue concerning the implementation of the Algorithm 1 in the form of a computer program. Steps 15 of Algorithm 1 should calculate the point of hyperdisk $D^{(k+1)}$ having the maximum bias. This task can be solved using the approach described in article [40]. It is based on the pseudoprojection operation, which is a generalization of the metric projection to a convex closed set. The idea of the solution is as follows. For the current approximation $\mathbf{u}$, let us find the numbers of all hyperplanes passing through $\mathbf{u}$:

$$\forall i \in \mathcal{H} : \langle \mathbf{a}_i,\mathbf{u}\rangle = b_i; \quad \forall i \in \mathcal{P}\backslash\mathcal{H} : \langle \mathbf{a}_i,\mathbf{u}\rangle \neq b_i.$$

Let $\mathfrak{H}$ be the set of all subsets of the set $\mathcal{H}$, except the empty set. This set defines a set of linear manifolds of the form

$$V(\mathcal{S}) = \bigcap_{i\in\mathcal{S}} H_i,$$

where $\mathcal{S}\in\mathfrak{H}$. Then, any face Q of the polytope M containing $\mathbf{u}$ can be represented as follows:

$$Q = M \cap V(\mathcal{S})$$

with $\mathcal{S}\in\mathfrak{H}$. Given the gradient $\mathbf{c}$ of the objective function, we can find the unit direction vector of maximum increase of the objective function for each face using the pseudoprojection operation. The face with the maximum increment will provide us with the direction vector d that will allow us to find the next approximation (see Fig. 2). We plan to describe this method in detail in a separate paper. However, we already have its parallel implementation in C++ called AlFaMove (Along Faces Movement) algorithm. Using AlFaMove, we conducted large-scale computational experiments on a cluster computing system, which showed high scalability of this algorithm. The source codes of AlFaMove are freely available on GitHub at https://github.com/leonid-sokolinsky/AlFaMove. All used test problems in MTX format [41] are available at https://github.com/leonid-sokolinsky/Set-of-LP-Problems.

The drawback of the AlFaMove algorithm is that its time complexity can be estimated as $O(2^m)$, where m is the number of constraints of the LP problem. We see the following way to solve this issue involving an artificial neural network. Using the approach described in paper [38], we replace the hyperdisk $D^{(k+1)}$ in Algorithm 1 with a set of points called the receptive field. We map each point of the receptive field to its bias relative to the boundary of the feasible polytope. As a result, we obtain a matrix of dimension $(n-1)$, which is a local image of the LP problem. The locality of the image means that we obtain a visual representation of the surface not of the entire feasible polytope, but only of some part of it in the neighborhood of the point of the current approximation. This image is fed to the input of a pre-trained feed forward neural network, which outputs the vector $\mathbf{d}$ indicating the direction of movement on the surface of the feasible polytope towards the maximum increase in the value of the objective function.

Algorithm 2 LP method using DNN

```
Require: $\hat{H}_i = \{\mathbf{x} \in \mathbb{R}^n | \langle \mathbf{a}_i, \mathbf{x} \rangle \leqslant b_i\}$; $\hat{M} = \bigcap_{i \in \mathcal{I}} \hat{H}_i$
 1: input $\mathbf{u}^{(0)}$
 2: assert $\mathbf{u}^{(0)} \in M \cap \Gamma(\hat{M})$
 3: $k := 0$
 4: $D^{(0)} := \mathfrak{G}(\mathbf{u}^{(0)})$
 5: $\mathbf{d}^{(0)} := \mathrm{DNN}(D^{(0)})$
 6: while $\mathbf{d}^{(k)} \neq \mathbf{0}$
 7:     $L^{(k)} = \{\mathbf{u}^{(k)} + \lambda \mathbf{d}^{(k)} \mid \lambda \in \mathbb{R}_{>0}\}$
 8:     $\mathbf{u}^{(k+1)} := \arg\max\{||\mathbf{x} - \mathbf{u}^{(k)}|| \mid \mathbf{x} \in L^{(k)} \cap \Gamma(M)\}$
 9:     $D^{(k+1)} := \mathfrak{G}(\mathbf{u}^{(k+1)})$
10:     $\mathbf{d}^{(k+1)} := \mathrm{DNN}(D^{(k+1)})$
11:     $k := k + 1$
12: end while
13: output $\mathbf{u}^{(k)}$
14: stop
```

Denote by $\mathfrak{G}(\mathbf{u})$ the function that constructs a receptive field centered at the point $\mathbf{u}$ and calculates the local image of the LP problem at this point. The algorithm for constructing a multidimensional image of the LP problem is described and investigated in [38]. Denote by DNN a deep neural network, which consumes a local image of the LP problem and outputs the vector $\mathbf{d}$ that defines the direction of movement along the surface of the feasible polytope. Thus, Algorithm 1 can be transformed into Algorithm 2. The set of labeled examples required for DNN training can be obtained using the AlFaMove algorithm described above. We plan to conduct a separate study specifically addressing this subject.

Let us estimate the time complexity of Algorithm 2. The surface movement method visits[4] each hyperplane of the recessive polytope no more than once. A visit to a hyperplane is performed within one iteration of the `while` loop (steps 6–12 of Algorithm 2). Therefore, the total number of iterations can be estimated as $O(m)$, where m is the number of constraints of LP Problem (1). Finding the next approximation $\mathbf{u}^{(k+1)}$ in Step 8 of Algorithm 2 can be implemented by dichotomy. Thus, the number of operations in Step 8 does not depend on the number of constraints m, nor on the dimension n, and, for large values of m and n, it can be estimated as a constant. The most time-consuming operation is the construction of a local image of LP Problem (1) in Step 9 of Algorithm 2. A receptive field in the form of a hypercubic lattice consists of $\eta^{(n-1)}$ points, where n is the dimension of space, and η is the number of points in one dimension. However, recent study [42] has shown that a cruciform receptive field with the number of points $(\eta - 1)n + 1$ gives results that are not inferior to the hypercubic one in terms of the accuracy of solving the LP problem. A pre-trained feedforward neural network DNN calculates the movement direction vector $\mathbf{d}$ at step 10 in a time that depends only on n, since it is fed $(\eta - 1)n + 1$ numbers (in the case of a cruciform receptive field). Thus, the total time complexity of the algorithm can be estimated as $O(mn)$.

In addition, we note that the scalability boundary[5] of the algorithm for constructing a visual image of the LP problem can be estimated as $O(\sqrt{2n^2m + m^2n + 8nm - 6m})$ [38]. Under the assumption that

[4]A visit to a hyperplane is understood as a rectilinear movement from the point of entry to the hyperplane to the point of the first change in the direction of movement.

[5]The scalability boundary refers to the number of processor nodes of a cluster computing system on which the maximum speedup is achieved.

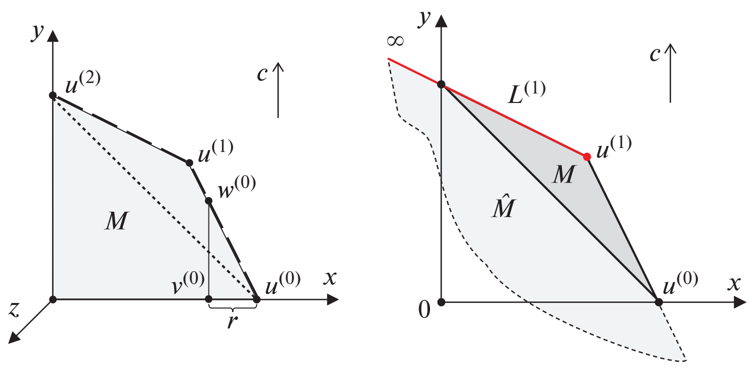

Fig. 4. Special cases of optimal objective path.

$m = O(n)$, we obtain, for the scalability boundary, the estimation $O(n\sqrt{n})$, which is close to a linear relationship. This means that the algorithm for constructing a local image of the LP problem can be effectively parallelized on a large number of processor nodes of a cluster computing system. So for $n = 7$ and $m = 15$, computational experiments show a peak of speedup on 326 processor nodes [38]. Note that the number of iterations of Algorithm 2 does not depend on ϵ_f, since there is no such parameter in this algorithm.

The surface movement method is self-correcting. Therefore, it can potentially be used to solve non-stationary problems. At that, if only the objective function changes, then Algorithm 2 does not require any crucial changes at all. It is important that the rate of correction outperforms the rate of change. If the system of constraints changes (without changing the dimension), then Algorithm 2 will require certain modifications, since the current approximation may "dive" into the polytope or "break away" from its surface. The authors intend to study this issue in detail in the future.

Algorithm 2 can also be used to solve LP problems in real time. Indeed, the number of iterations is bounded by the parameter m. With a fixed n, building a local image of the LP problem requires a fixed number of operations. At the same time, the image construction procedure is effectively parallelized on a large number of processor nodes. The feedforward neural network DNN processes the local image of the LP problem in a fixed time, depending only on n. The work of a neural network can also be efficiently parallelized using GPUs. In the future, it is possible to refuse the traveling along the faces/edges of the feasible polytope and analyze with the use of a neural network the image of the entire feasible polytope obtained from the apex point (see [40]). The neural network will produce an approximate solution, which can be refined by analyzing a fixed number of local images with increasing detail. However, this issue needs further research.

The Algorithm 1 constructs the optimal objective path to a solution of the LP problem on the surface of the feasible polytope. This directly follows from the construction of the algorithm and Proposition 3. An interesting question is whether this path is always the path of the shortest length in the sense of the Euclidean metric. The following simple example in the space $\mathbb{R}^3$ shows that this is not always the case. For the system of constraints

$$x + 2y \leqslant 2, \quad 2x + y \leqslant 2, \quad x \geqslant 0, \quad y \geqslant 0, \quad z = 0;$$

and the objective function $f(x, y, z) = y$, the optimal objective path may not coincide with the shortest path (see Fig. 4a).

The question may also arise, whether it is possible to replace $\Gamma(M)$ with $\Gamma(\hat{M})$ in Step 13 of Algorithm 1, since this reduces the number of inequalities used for checking the condition $x \in \Gamma(M)$. The answer is negative. For example (see Fig. 4b), in the space $\mathbb{R}^2$, for the system of constraints

$$x + 2y \leqslant 2, \quad 2x + y \leqslant 2, \quad x + y \geqslant 1, \quad x \geqslant 0, \quad y \geqslant 0;$$

and the objective function $f(x, y) = y$, with $\mathbf{u}^{(1)} = (2/3, 2/3)$, we obtain

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

As a drawback of the surface movement method, it can be noted that it is not affine invariant, since the cost functional is identified with the vector. Thus, the behavior of the method depends on the Euclidean structure defined by the coordinate system.

The scientific contribution and theoretical significance of the proposed method is that, for the first time, it opens up the possibility of using feedforward neural networks to solve multidimensional LP problems based on the analysis of their images.

5. CONCLUSIONS

The article presented a new method for solving the linear programming problem (LP), called the "surface movement method". This method constructs, on the surface of the polytope bounding the feasible region, a path from an initial point to a point of solving the LP problem. The movement vector is always constructed in the direction of the maximum increase/decrease in the value of the objective function. The resulting path is called the optimal objective path.

The surface movement method assumes the use of a feedforward neural network to determine the direction of movement along the faces of the feasible polytope. To do this, a local image of the multidimensional LP problem is constructed at the point of the current approximation, which is fed to the input of the neural network. The set of labeled precedents needed for training a neural network can be obtained using the apex method.

To build a theoretical basis of the surface movement method, the concept of the objective projection is introduced. The objective projection is an oblique projection in the direction parallel to the gradient vector of the objective function. A scalar quantity called bias is defined. The bias modulus is equal to the distance from the point to its objective projection. The bias sign is determined by the position of the point inside or outside of the feasible polytope. An equation for calculating the bias of a point relative to the boundary of the feasible polytope is obtained. It is shown that a larger bias corresponds to a larger value of the objective function. A formalized description of the surface movement method in the form of an algorithm is presented. The main convergence theorem of the surface movement method to the solution of the LP problem in a finite number of iterations is proved. A version of the surface movement algorithm using a function of constructing a local multidimensional image of the LP problem and a deep neural network is provided.

As directions for further research, the following can be indicated.

1. Design and training of a DNN network capable of calculating the movement vector in the direction of maximizing the value of the objective function for multidimensional LP problems.

2. Development of a software package for a cluster computing system implementing Algorithm 2 by combining supercomputer and neural network technologies.

3. Study of the dependence of the DNN network accuracy on the density of the receptive field.

4. Study of the suitability of the surface movement method for solving non-stationary LP problems.

5. Study of the suitability of the surface movement method for solving LP problems in real time.

6. Development of a new visual method for solving LP problems using neural networks based on the analysis of the image of the feasible polytope as a whole.

Appendix

NOTATIONS

$\mathbb{R}^n$	real Euclidean space
$\|\cdot\|$	Euclidean norm
$\langle\cdot,\cdot\rangle$	dot product of two vectors
$f(\mathbf{x})$	linear objective function
$\mathbf{c}$	gradient of objective function $f(\mathbf{x})$
$\mathbf{e_c}$	unit vector parallel to vector $\mathbf{c}$
$\bar{\mathbf{x}}$	solution of LP problem
$\mathbf{a}_i$	ith row of matrix A
H_i	hyperplane defined by equation $\langle\mathbf{a}_i,\mathbf{x}\rangle = b_i$
$\hat{H}_i$	half-space defined by inequality $\langle\mathbf{a}_i,\mathbf{x}\rangle \leqslant b_i$
$\mathcal{P}$	set of row indexes of matrix A
$\mathcal{I}$	index set of recessive half-spaces
M	feasible polytope
$\hat{M}$	recessive polytope
$\Gamma(M)$	boundary of M
$\Gamma(\hat{M})$	boundary of $\hat{M}$
$\boldsymbol{\gamma}_i(\mathbf{z})$	objective projection of $\mathbf{z}$ onto H_i
$\beta_i(\mathbf{z})$	bias of $\mathbf{z}$ relative to H_i
$\hat{\boldsymbol{\gamma}}(\mathbf{z})$	objective projection of $\mathbf{z}$ onto $\Gamma(\hat{M})$
$\hat{\beta}(\mathbf{z})$	bias of $\mathbf{z}$ relative to $\Gamma(\hat{M})$
$V_r(\mathbf{x})$	n-dimensional ball of radius r centered at point $\mathbf{x}$

FUNDING

The research was supported by the Russian Science Foundation (project no. 23-21-00356).

CONFLICT OF INTEREST

The authors of this work declare that they have no conflicts of interests.

REFERENCES

1. T. Hartung, "Making big sense from big data," Front. Big Data **1** (5), 1–2 (2018). https://doi.org/10.3389/fdata.2018.00005
2. I. M. Sokolinskaya and L. B. Sokolinsky, "On the solution of linear programming problems in the age of big data," in *Parallel Computational Technologies PCT 2017,* Vol. 753 of *Communications in Computer and Information Science,* Ed. by L. Sokolinsky and M. Zymbler (Springer, Cham, Switzerland, 2017), pp. 86–100. https://doi.org/10.1007/978-3-319-67035-5_7

3. M. H. Veatch, *Linear and Convex Optimization: A Mathematical Approach* (Wiley, Hoboken, NJ, 2021). https://doi.org/10.1002/9781119664079
4. J. Branke, "Optimization in dynamic environments," in *Evolutionary Optimization in Dynamic Environments. Genetic Algorithms and Evolutionary Computation* (Springer, Boston, MA, 2002), Vol. 3, pp. 13–29. https://doi.org/10.1007/978-1-4615-0911-0_2
5. I. I. Eremin and V. D. Mazurov, *Nonstationary Processes of Mathematical Programming* (Nauka, Moscow, 1979) [in Russian].
6. J. Brogaard, T. Hendershott, and R. Riordan, "High-frequency trading and price discovery," Rev. Financ. Stud. **27**, 2267–2306 (2014). https://doi.org/10.1093/rfs/hhu032
7. A. Tewari, "Optimal navigation and control of aircraft," in *Advanced Control of Aircraft, Spacecraft and Rockets* (Wiley, Noida, India, 2011), Chap. 3, pp. 103–194. https://doi.org/10.1002/9781119971191.ch3
8. B. Srinivasan, S. Palanki, and D. Bonvin, "Dynamic optimization of batch processes. I. Characterization of the nominal solution," Comput. Chem. Eng. **27**, 1–2 (2003). https://doi.org/10.1016/S0098-1354(02)00116-3
9. J. Fleming, X. Yan, C. Allison, N. Stanton, and R. Lot, "Real-time predictive eco-driving assistance considering road geometry and long-range radar measurements," IET Intell. Transp. Syst. **15**, 573–583 (2021). https://doi.org/10.1049/ITR2.12047
10. M. Scholl, K. Minnerup, C. Reiter, B. Bernhardt, E. Weisbrodt, and S. Newiger, "Optimization of a thermal management system for battery electric vehicles," in *Proceedings of the 14th International Conference on Ecological Vehicles and Renewable Energies, EVER 2019* (IEEE, Monaco, 2019), pp. 1–10. https://doi.org/10.1109/EVER.2019.8813657
11. S. Meisel, "Dynamic vehicle routing," in *Anticipatory Optimization for Dynamic Decision Making,* Vol. 51 of *Operations Research/Computer Science Interfaces Series* (Springer, New York, 2011), pp. 77–96. https://doi.org/10.1007/978-1-4614-0505-4_6
12. D. Kiran, *Production Planning and Control: A Comprehensive Approach* (Elsevier, Oxford, 2019). https://doi.org/10.1016/C2018-0-03856-6
13. R. Mall, *Real-Time Systems: Theory and Practice* (Pearson Education, Delhi, India, 2007). https://doi.org/10.1007/978-1-4614-0505-4_6
14. G. B. Dantzig, *Linear Programming and Extensions* (Princeton Univ. Press, Princeton, NJ, 1998).
15. J. Hall and K. McKinnon, "Hyper-sparsity in the revised simplex method and how to exploit it," Comput. Optim. Appl. **32**, 259–283 (2005). https://doi.org/10.1007/s10589-005-4802-0
16. Y. Disser, O. Friedmann, and A. V. Hopp, "An exponential lower bound for Zadeh's pivot rule," Math. Program. **199**, 865–936 (2023). https://doi.org/10.1007/s10107-022-01848-x
17. A. V. Hopp, *The Complexity of Zadeh's Pivot Rule* (Logos, Berlin, 2020). https://doi.org/10.30819/5206
18. V. Klee and G. J. Minty, "How good is the simplex algorithm?" in *Inequalities III, Proceedings of the 3rd Symposium on Inequalities, Los Angeles, Sept. 1–9, 1969,* Ed. by O. Shisha (Academic, New York, 1972), pp. 159–175.
19. R. Bartels, J. Stoer, and C. Zenger, "A realization of the simplex method based on triangular decompositions," in *Handbook for Automatic Computation,* Vol. 2: *Linear Algebra* (Springer, Berlin, 1971), pp. 152–190. https://doi.org/10.1007/978-3-642-86940-2_11
20. P. Tolla, "A survey of some linear programming methods," in *Concepts of Combinatorial Optimization,* Ed. by V. T. Paschos, 2nd ed. (Wiley, Hoboken, NJ, 2014), Chap. 7, pp. 157–188. https://doi.org/10.1002/9781119005216.ch7
21. J. Hall, "Towards a practical parallelisation of the simplex method," Comput. Manage. Sci. **7**, 139–170 (2010). https://doi.org/10.1007/s10287-008-0080-5
22. B. Mamalis and G. Pantziou, "Advances in the parallelization of the simplex method," in *Algorithms, Probability, Networks, and Games,* Lect. Notes Comput. Sci. **9295**, 281–307 (2015). https://doi.org/10.1007/978-3-319-24024-4_17
23. V. Zorkaltsev and I. Mokryi, "Interior point algorithms in linear optimization," J. Appl. Ind. Math. **12**, 191–199 (2018). https://doi.org/10.1134/S1990478918010179
24. G. B. Dantzig and M. N. Thapa, "Early interior-point methods," in *Linear Programming 2: Theory and Extensions. Springer Series in Operations Research and Financial Engineering* (Springer, New York, 2003), Chap. 3, pp. 67–122. https://doi.org/10.1007/0-387-21569-7_3
25. I. Dikin, "Iterative solution of problems of linear and quadratic programming," Sov. Math. Dokl. **8**, 674–675 (1967).
26. J. Gondzio, "Interior point methods 25 years later," Eur. J. Oper. Res. **218**, 587–601 (2012). https://doi.org/10.1016/j.ejor.2011.09.017

27. C. Roos, T. Terlaky, and J.-P. Vial, *Interior Point Methods for Linear Optimization* (Springer, New York, 2005). https://doi.org/10.1007/b100325
28. I. M. Sokolinskaya, "Parallel method of pseudoprojection for linear inequalities," in *Parallel Computational Technologies. PCT 2018,* Vol. 910 of *Communications in Computer and Information Science,* Ed. by L. Sokolinsky and M. Zymbler (Springer, Cham, Switzerland, 2018), pp. 216–231. https://doi.org/10.1007/978-3-319-99673-8_16
29. J. Gondzio and A. Grothey, "Direct solution of linear systems of size 109 arising in optimization with interior point methods," in *Parallel Processing and Applied Mathematics, PPAM 2005,* Lect. Notes Comput. Sci. **3911**, 513–525 (2006). https://doi.org/10.1007/11752578_62
30. A. Prieto, B. Prieto, E. M. Ortigosa, E. Ros, F. Pelayo, J. Ortega, and I. Rojas, "Neural networks: An overview of early research, current frameworks and new challenges," Neurocomputing **214**, 242–268 (2016). https://doi.org/10.1016/j.neucom.2016.06.014
31. D. W. Tank and J. J. Hopfield, "Simple 'neural' optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit," IEEE Trans. Circuits Syst. **33**, 533–541 (1986). https://doi.org/10.1109/TCS.1986.1085953
32. M. P. Kennedy and L. O. Chua, "Unifying the Tank and Hopfield linear programming circuit and the canonical nonlinear programming circuit of Chua and Lin," IEEE Trans. Circuits Syst. **34**, 210–214 (1987). https://doi.org/10.1109/TCS.1987.1086095
33. X. Liu and M. Zhou, "A one-layer recurrent neural network for non-smooth convex optimization subject to linear inequality constraints," Chaos, Solitons Fract. **87**, 39–46 (2016). https://doi.org/10.1016/j.chaos.2016.03.009
34. A. Malek and A. Yari, "Primal-dual solution for the linear programming problems using neural networks," Appl. Math. Comput. **167**, 198–211 (2005). https://doi.org/10.1016/J.AMC.2004.06.081
35. A. Rodriguez-Vazquez, R. Dominguez-Castro, A. Rueda, J. L. Huertas, and E. Sanchez-Sinencio, "Nonlinear switched-capacitor 'Neural' networks for optimization problems," IEEE Trans. Circuits Syst. **37**, 384–398 (1990). https://doi.org/10.1109/31.52732
36. S. H. Zak and V. Upatising, "Solving linear programming problems with neural networks: A comparative study," IEEE Trans. Neural Networks **6**, 94–104 (1995). https://doi.org/10.1109/72.363446
37. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature (London, U.K.) **521** (7553), 436–444 (2015). https://doi.org/10.1038/nature14539
38. N. Olkhovsky and L. Sokolinsky, "Visualizing multidimensional linear programming problems," in *Parallel Computational Technologies PCT 2022,* Vol. 1618 of *Communications in Computer and Information Science,* Ed. by L. Sokolinsky and M. Zymbler (Springer, Cham, Switzerland, 2022), pp. 172–196. https://doi.org/10.1007/978-3-031-11623-0_13
39. R. Raina, A. Madhavan, and A. Y. Ng, "Large-scale deep unsupervised learning using graphics processors," in *Proceedings of the 26th Annual International Conference on Machine Learning ICML'09* (ACM, New York, 2009), pp. 873–880. https://doi.org/10.1145/1553374.1553486
40. L. B. Sokolinsky and I. M. Sokolinskaya, "Apex method: A new scalable iterative method for linear programming," Mathematics **11** (7), 1–28 (2023). https://doi.org/Sokolinsky2023:TR
41. R. F. Boisvert, R. Pozo, and K. A. Remington, "The matrix market exchange formats: Initial design," Tech. Report NISTIR 5935 (Natl. Inst. Stand. Technol., Gaithersburg, MD, 1996). https://nvlpubs.nist.gov/nistpubs/Legacy/IR/nistir5935.pdf. Accessed June 26, 2024.
42. N. Olkhovsky, "Study of the receptive field structure in the visual method of solving the linear programming problem," Preprint (2023). https://doi.org/10.24108/preprints-3112771