On the Solution of Linear Programming Problems in the Age of Big Data

Irina Sokolinskaya and Leonid B. Sokolinsky(✉)

South Ural State University, 76 Lenin Prospekt, Chelyabinsk 454080, Russia
{Irina.Sokolinskaya,Leonid.Sokolinsky}@susu.ru

Abstract. The Big Data phenomenon has spawned large-scale linear programming problems. In many cases, these problems are non-stationary. In this paper, we describe a new scalable algorithm called *NSLP* for solving high-dimensional, non-stationary linear programming problems on modern cluster computing systems. The algorithm consists of two phases: *Quest* and *Targeting*. The *Quest* phase calculates a solution of the system of inequalities defining the constraint system of the linear programming problem under the condition of dynamic changes in input data. To this end, the apparatus of Fejer mappings is used. The *Targeting* phase forms a special system of points having the shape of an n-dimensional axisymmetric cross. The cross moves in the n-dimensional space in such a way that the solution of the linear programming problem is located all the time in an ε-vicinity of the central point of the cross.

Keywords: *NSLP* algorithm · Non-stationary linear programming problem · Large-scale linear programming · Fejer mapping

1 Introduction

The Big Data phenomenon has spawned large-scale linear programming (LP) problems [1]. Such problems arise in many different fields. In [2], the following large-scale industrial optimization problems are presented within the context of big data:

- schedule crews for 3400 daily flights in 40 countries;
- buy ads in 10–15 local publications across 40 000 zip codes;
- pick one of 742 trillion choices in creating the US National Football League schedule;
- select 5 offers out of 1000 for each of 25 000 000 customers of an online store;
- place 1000 stock keeping units on dozens of shelves in 2000 stores;
- decide among 200 000 000 maintenance routing options.

The reported study has been partially supported by the RFBR according to research project No. 17-07-00352-a, by the Government of the Russian Federation according to Act 211 (contract No. 02.A03.21.0011) and by the Ministry of Education and Science of the Russian Federation (government order 2.7905.2017/8.9).

L. Sokolinsky and M. Zymbler (Eds.): PCT 2017, CCIS 753, pp. 86–100, 2017.
DOI: 10.1007/978-3-319-67035-5_7

Each of these problems uses Big Data from the subject field. Such a problem is formalized as a linear programming problem involving up to tens of millions of constraints and up to hundreds of millions of decision variables.

Gondzio [3] presents a certain class of large-scale optimization problems arising in quantum information science and related to Bell's theorem. These problems are two-level optimization problems. The higher-level problem is a non-convex non-linear optimization task. It requires solving hundreds of linear programming problems, each of which can contain millions of constraints and millions of variables.

Mathematical modeling in economics is another source of large-scale LP problems. In many cases, LP problems arising in mathematical economy are non-stationary (dynamic). For example, Sodhi [4] describes a dynamic LP task for asset-liability management. This task involves 1.7 billion constraints and 5.1 billion variables. Algorithmic trading is another area that generates large-scale non-stationary linear programming problems [5–7]. In such problems, the number of variables and inequalities in the constraint system formed by using Big Data can reach tens and even hundreds of thousands, and the period of input data change is within the range of hundredths of a second.

Until now, one of the most popular methods for solving LP problems is the class of algorithms proposed and designed by Dantzig on the basis of the simplex method [8]. The simplex method has proved to be effective in solving a large class of LP problems. However, Klee and Minty [9] gave an example showing that the worst-case complexity of the simplex method is exponential time. Nevertheless, Khaciyan [10] proved that the LP problem can be solved in polynomial time by a variant of an iterative ellipsoidal algorithm developed by Shor [11]. Attempts to apply the ellipsoidal algorithm in practice have been unsuccessful so far. In most cases, this algorithm demonstrated much worse efficiency than the simplex method did. Karmarkar [12] proposed the *interior-point method*, which runs in polynomial time and is also very efficient in practice.

The simplex method and the interior-point method remain today the main methods for solving the LP problem. However, these methods may prove ineffective in the case of large-scale LP problems with rapidly changing and (or) incomplete input data. The authors described in [13] a parallel algorithm for solving LP problems with non-formalized constraints. The main idea of the proposed approach is to combine linear programming and discriminant analysis methods. Discriminant analysis requires two sets of patterns M and N. The first set must satisfy the non-formalized constraints, while the second must not. To obtain representative patterns, methods of data mining [14] and time series analysis can be used [15]. To overcome the problem of non-stationary input data, the authors proposed in [16,17] the pursuit algorithm for solving non-stationary LP problems on cluster computing systems. The pursuit algorithm uses Fejer mappings (see [18]) to build a pseudo-projection onto a convex bounded set. The pseudo-projection operator is similar to a projection, but in contrast to the last, it is stable to dynamic changes in input data. In [19], the authors investigated the efficiency of using Intel Xeon Phi multi-core processors to calculate the pseudo-projections.

In this paper, we describe the new *NSLP* (Non-Stationary Linear Programming) algorithm for solving large-scale non-stationary LP problems on cluster computing systems. The *NSLP* algorithm is more efficient than the pursuit algorithm, since it uses a compute-intensive pseudo-projection operation only once (the pursuit algorithm computes pseudo-projections K times at each iteration, K being the number of processor nodes). The rest of the paper is organized as follows. Section 2 gives a formal statement of an LP problem and presents the definitions of the Fejer process and the pseudo-projection onto a polytope. Section 3 describes the new *NSLP* algorithm. Section 4 summarizes the obtained results and proposes directions for future research.

2 Problem Statement

Let a non-stationary LP problem be given in the vector space $\mathbb{R}^n$:

$$\max\left\{\langle c_t, x\rangle \,|A_t x \le b_t,\ x \ge 0\right\}, \tag{1}$$

where the matrix A_t has m rows. The non-stationarity of the problem means that the values of the elements of the matrix A_t and the vectors b_t, c_t depend on time $t \in \mathbb{R}_{\ge 0}$. We assume that the value of $t = 0$ corresponds to the initial time:

$$A_0 = A, b_0 = b, c_0 = c. \tag{2}$$

Let us define the map $\varphi_t \colon \mathbb{R}^n \to \mathbb{R}^n$ as follows:

$$\varphi_t(x) = x - \frac{\lambda}{m}\sum_{i=1}^{m} \frac{\max\left\{\langle a_{ti}, x\rangle - b_{ti}, 0\right\}}{\|a_{ti}\|^2} \cdot a_{ti}, \tag{3}$$

where a_{ti} is the i-th row of the matrix A_t, and $b_{t1}, \ldots, b_{tm}$ are the elements of the column b_t. Let us denote

$$\varphi(x) = \varphi_0(x) = x - \frac{\lambda}{m}\sum_{i=1}^{m} \frac{\max\left\{\langle a_i, x\rangle - b_i, 0\right\}}{\|a_i\|^2} \cdot a_i. \tag{4}$$

Let M_t be the polytope defined by the constraints of the non-stationary LP problem (1). Such a polytope is always convex. It is known (see [18]) that φ_t is a continuous single-valued M_t-fejerian[1] map for the relaxation factor $0 < \lambda < 2$.

By definition, put

$$\varphi_t^s(x) = \underbrace{\varphi_t \ldots \varphi_t(x)}_{s}. \tag{5}$$

[1] A single-valued map $\varphi \colon \mathbb{R}^n \to \mathbb{R}^n$ is said to be *fejerian* relatively to a set M (or briefly, *M-fejerian*) if

$$\varphi(y) = y, \forall y \in M;$$
$$\|\varphi(x) - y\| < \|x - y\|, \forall x \notin M, \forall y \in M.$$

.

The *Fejer process* generated by the map φ_t for an arbitrary initial approximation $x_0 \in \mathbb{R}^n$ is the sequence $\{\varphi_t^s(x_0)\}_{s=0}^{+\infty}$. It is known (see Lemma 39.1 in [20]) that the Fejer process for a fixed t converges to a point belonging to the polytope M_t:

$$\{\varphi_t^s(x_0)\}_{s=0}^{+\infty} \to \bar{x} \in M_t. \tag{6}$$

Let us consider the simplest non-stationary case, which is a translation of the polytope $M = M_0$ by the fixed vector $d \in \mathbb{R}^n$ in one unit of time. In this case, $A_t = A, c_t = c$, and the non-stationary problem (1) takes the form

$$\max\{\langle c, x\rangle \,|\, A(x - td) \le b,\ x \ge 0\}, \tag{7}$$

which is equivalent to

$$\max\{\langle c, x\rangle \,|\, Ax \le b + Atd,\ x \ge 0\}.$$

Comparing this with (1), we obtain $b_t = b + Atd$. In this case, the M_t-fejerian map (3) is converted to the following:

$$\varphi_t(x) = x - \frac{\lambda}{m}\sum_{i=1}^{m} \frac{\max\{\langle a_i, x\rangle - (b_i + \langle a_i, td\rangle), 0\}}{\|a_i\|^2} \cdot a_i,$$

which is equivalent to

$$\varphi_t(x) = x - \frac{\lambda}{m}\sum_{i=1}^{m} \frac{\max\{\langle a_i, x - td\rangle - b_i, 0\}}{\|a_i\|^2} \cdot a_i \tag{8}$$

The *φ-projection* (*pseudo-projection*) of the point $x \in \mathbb{R}^n$ on the polytope M is the map $\pi_M^{\varphi}(x) = \lim_{s\to\infty} \varphi^s(x)$.

3 The *NSLP* Algorithm

The *NSLP* (*Non-Stationary Linear Programming*) algorithm is designed to solve large-scale non-stationary LP problems on cluster computing systems. It consists of two phases: *Quest* and *Targeting*. The *Quest* phase calculates a solution of the system of inequalities defining the constraint system of the linear programming problem under the condition of dynamic changes in input data. To this end, the apparatus of Fejer mappings is used. The *Targeting* phase forms a special system of points having the shape of an n-dimensional axisymmetric cross. The cross moves in the n-dimensional space in such a way that the solution of the LP problem remains permanently in an ε-vicinity of the central point of the cross. Let us describe both phases of the algorithm in more detail.

3.1 The *Quest* Phase

Without loss of generality, we can assume that all the calculations are performed in the region of positive coordinates. At the beginning, we choose an arbitrary point $z_0 \in \mathbb{R}^n_{\geq 0}$ with non-negative coordinates. This point plays the role of initial approximation for the problem (1). Then we organize an iterative Fejer process of the form (6). During this process, the Fejer approximations are consecutively calculated by using the Fejer mapping (3). This process converges to a point located on the polytope M_t. Owing to the non-stationary nature of the problem (1), the polytope M_t can change its position and shape during the calculation of the pseudo-projection. An input data update is performed every L iterations, L being some fixed positive integer that is a parameter of the algorithm. Let us denote by $t_0, t_1, \ldots, t_k, \ldots$ sequential time points corresponding to the instants of input data update. Without loss of generality, we can assume that

$$t_0 = 0, t_1 = L, t_2 = 2L, \ldots, t_k = kL, \ldots. \tag{9}$$

This corresponds to the case when one unit of time is equal to the time spent by the computer to calculate one value of the Fejer mapping using Eq. (3).

Let the polytope M_t take shapes and locations

$$M_0, M_1, \ldots, M_k, \ldots$$

at time points (9). Let

$$\varphi_0, \varphi_1, \ldots, \varphi_k, \ldots$$

be the Fejer mappings determined by Eq. (3) taking into account the changes in input data of problem (1) at time points (9). In the *Quest* phase, the iterative process calculates the following sequence of points (see Fig. 1):

$$\{z_1 = \varphi_0^L(z_0), z_2 = \varphi_1^L(z_1), \ldots, z_k = \varphi_{k-1}^L(z_{k-1}), \ldots\}.$$

Let us briefly denote this iterative process as

$$\left\{\varphi_k^L(z_0)\right\}_{k=0}^{+\infty}. \tag{10}$$

It terminates when[2]

$$\operatorname{dist}\left(\varphi_k^L(z_{k-1}), M_k\right) < \varepsilon,$$

where $\varepsilon > 0$ is a positive real number being a parameter of the algorithm. One of the most important issues is the convergence of the iterative process (10). In the general case, this issue remains open. However, the following theorem holds for the non-stationary problem (7).

[2] Here $\operatorname{dist}(z, M) = \inf\{\|z - x\| : x \in M\}$.

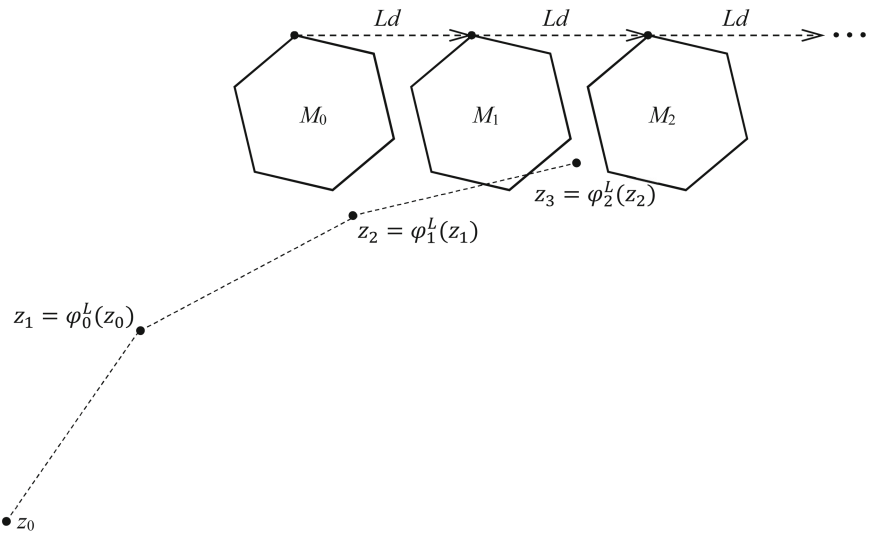

Fig. 1. The iterative process in the *Quest* phase for problem (7)

Theorem 1. *Let a non-stationary LP problem be given by (7). Let the Fejer mappings* $\varphi_0, \varphi_1, \ldots, \varphi_k, \ldots$ *be defined by the equation*

$$\varphi_k(x) = x - \frac{\lambda}{m} \sum_{i=1}^{m} \frac{\max\{\langle a_i, x - kLd\rangle - b_i, 0\}}{\|a_i\|^2} \cdot a_i. \tag{11}$$

This equation is derived using (8) and (9). By definition, put

$$z_k = \varphi_{k-1}^L(z_{k-1}) \tag{12}$$

where $k = 1, 2, \ldots$. *Then*

$$\lim_{k\to\infty} \operatorname{dist}(z_k, M_k) = 0 \tag{13}$$

under the following condition:

$$\forall x \in \mathbb{R}^n \backslash M \left(\|Ld\| < \operatorname{dist}(x, M) - \operatorname{dist}(\varphi^L(x), M)\right). \tag{14}$$

The Theorem 1 gives a sufficient condition for the convergence of the iterative process shown in Fig. 1. To prove this theorem, we will need the following auxiliary lemma.

Lemma 1. *Under the conditions of Theorem 1, we have*

$$v - u = pLd \Rightarrow \varphi_p^l(v) - \varphi^l(u) = pLd \tag{15}$$

for any $p = 0, 1, 2, \ldots$, $l = 1, 2, 3, \ldots$ *and* $u, v \in \mathbb{R}^n$.

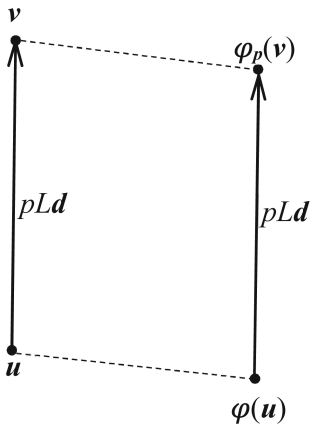

Fig. 2. Illustration to the proof of Lemma 1

Proof. The proof is by induction on l.

Induction base. Let $l = 1$, then the following condition holds:

$$v - u = pLd. \tag{16}$$

Then using (4), (11) and (16), we get

$$\begin{aligned}
\varphi_p(v) - \varphi(u) &= \varphi_p(u + pLd) - \varphi(u) \\
&= u + pLd - \frac{\lambda}{m} \sum_{i=1}^{m} \frac{\max\{\langle a_i, u\rangle - b_i, 0\}}{\|a_i\|^2} \cdot a_i - \varphi(u) \\
&= u + pLd - \frac{\lambda}{m} \sum_{i=1}^{m} \frac{\max\{\langle a_i, u\rangle - b_i, 0\}}{\|a_i\|^2} \cdot a_i \\
&\quad - u + \frac{\lambda}{m} \sum_{i=1}^{m} \frac{\max\{\langle a_i, u\rangle - b_i, 0\}}{\|a_i\|^2} \cdot a_i = pLd.
\end{aligned}$$

Thus, (15) holds if $l = 1$ (see Fig. 2).

Inductive step. Assume that condition (16) is true. Using the induction hypothesis, we get

$$\varphi_p^{l-1}(v) - \varphi^{l-1}(u) = pLd. \tag{17}$$

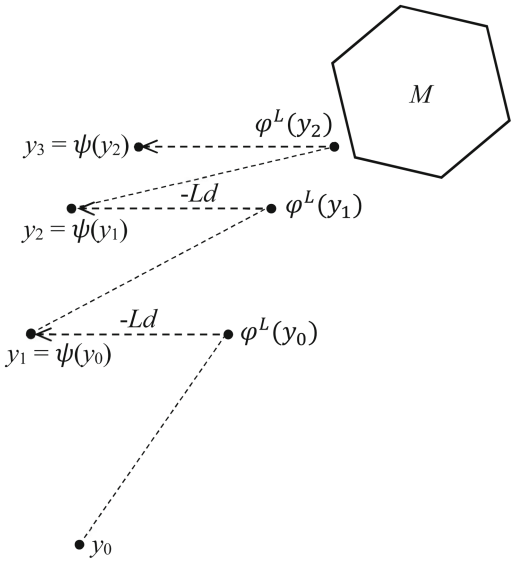

Fig. 3. The process defined by (18)

Then, combining (4), (5), (17) and (11), we obtain

$$\begin{aligned}
\varphi_p^l(v) - \varphi^l(u) &= \varphi_p(\varphi_p^{l-1}(v)) - \varphi(\varphi^{l-1}(u)) \\
&= \varphi_p(\varphi^{l-1}(u) + pLd) - \varphi(\varphi^{l-1}(u)) \\
&= \varphi^{l-1}(u) + pLd - \frac{\lambda}{m}\sum_{i=1}^{m} \frac{\max\left\{\langle a_i, \varphi^{l-1}(u)\rangle - b_i, 0\right\}}{\|a_i\|^2} \cdot a_i - \varphi(\varphi^{l-1}(u)) \\
&= \varphi^{l-1}(u) + pLd - \frac{\lambda}{m}\sum_{i=1}^{m} \frac{\max\left\{\langle a_i, \varphi^{l-1}(u)\rangle - b_i, 0\right\}}{\|a_i\|^2} \cdot a_i \\
&\quad - \varphi^{l-1}(u) + \frac{\lambda}{m}\sum_{i=1}^{m} \frac{\max\left\{\langle a_i, \varphi^{l-1}(u)\rangle - b_i, 0\right\}}{\|a_i\|^2} \cdot a_i = pLd.
\end{aligned}$$

This completes the proof of Lemma 1.

Proof (of Theorem 1). Let us fix an arbitrary point $z_0 \in \mathbb{R}^n \backslash M$. Let the map $\psi\colon \mathbb{R}^n \to \mathbb{R}^n$ be given by

$$\begin{aligned}
&\psi(x) = \varphi^L(x) - Ld, \forall x \notin M; \\
&\psi(x) = x, \forall x \in M.
\end{aligned} \tag{18}$$

By definition, put

$$y_0 = z_0 \tag{19}$$

and

$$y_k = \psi(y_{k-1}) \tag{20}$$

for $k = 1, 2, \ldots$ (see Fig. 3).

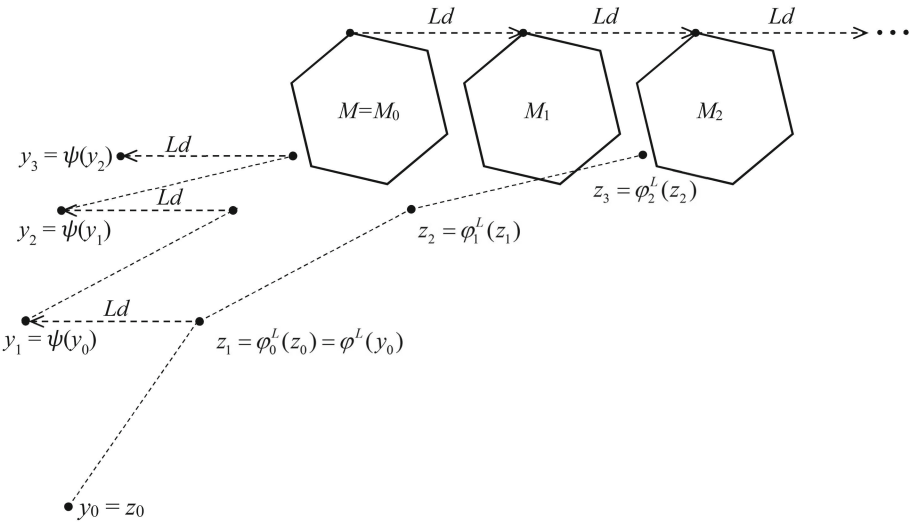

Fig. 4. Illustration to Eq. (21)

Now let us show by induction on k that

$$z_k - y_k = kLd \tag{21}$$

for $k = 0, 1, 2, \ldots$ (see Fig. 4).

Induction base. Equation (21) holds for $k = 0$. Taking into account (19), we see that the equation

$$z_0 - y_0 = 0 \cdot Ld$$

holds.

Inductive step. Suppose that

$$z_{k-1} - y_{k-1} = (k-1)Ld \tag{22}$$

for $k > 0$. Substituting $u = y_{k-1}, v = z_{k-1}, l = L, p = k-1$ in Lemma 1, and using (15), we obtain

$$z_{k-1} - y_{k-1} = (k-1)Ld \Rightarrow \varphi^L_{k-1}(z_{k-1}) - \varphi^L(y_{k-1}) = (k-1)Ld.$$

Comparing this with (22), we have

$$\varphi^L_{k-1}(z_{k-1}) - \varphi^L(y_{k-1}) = (k-1)Ld. \tag{23}$$

Combining (12), (18), (20) and (23), we get

$$\begin{aligned} z_k - y_k &= z_k - \psi(y_{k-1}) = z_k - \varphi^L(y_{k-1}) + Ld \\ &= \varphi^L_{k-1}(z_{k-1}) - \varphi^L(y_{k-1}) + Ld = (k-1)Ld + Ld = kLd. \end{aligned}$$

Thus, Eq. (21) holds.

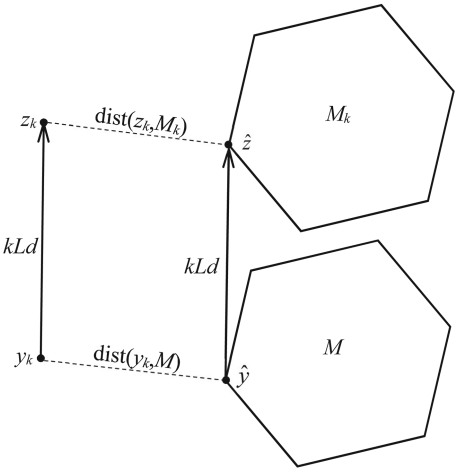

Fig. 5. Illustration to Eq. (24)

Now we show that

$$\operatorname{dist}(z_k, M_k) = \operatorname{dist}(y_k, M) \tag{24}$$

for all $k = 0, 1, 2, \ldots$. Let us choose a point $\hat{y} \in M$ that satisfies the following condition:

$$\|\hat{y} - y_k\| = \operatorname{dist}(y_k, M). \tag{25}$$

Such a point exists and is unique since the polytope M is a bounded, closed and convex set. The polytope M_k is the result of translating the polytope M by the vector kLd (see Fig. 5). Since $\hat{y} \in M$, it follows that the point $\hat{z} = \hat{y} + kLd$ belongs to the polytope M_k. Taking into account (21), we conclude that the points $\{y_k, z_k, \hat{z}, \hat{y}\}$ are the vertices of a parallelogram. Therefore,

$$\|\hat{z} - z_k\| = \|\hat{y} - y_k\|. \tag{26}$$

Let us show that

$$\|\hat{z} - z_k\| = \operatorname{dist}(z_k, M_k). \tag{27}$$

Assume for a contradiction that $\exists z' \in M_k$ such that

$$\|z' - z_k\| < \|\hat{z} - z_k\|. \tag{28}$$

Since $z' \in M_k$, it follows that the point $y' = z' - kLd$ belongs to the polytope M. Now, if we recall that the points $\{y_k, z_k, \hat{z}, \hat{y}\}$ are the vertices of a parallelogram, we get

$$\|y' - y_k\| = \|z' - z_k\|.$$

Combining this with (25), (26) and (28), we obtain

$$\|y' - y_k\| = \|z' - z_k\| < \|\hat{z} - z_k\| = \|\hat{y} - y_k\| = \operatorname{dist}(y_k, M).$$

It follows that

$$\exists y' \in M \left(\|y' - y_k\| < \operatorname{dist}(y_k, M) \right).$$

This contradicts the definition of the distance between a point and a set. Therefore, Eq. (27) holds. Combining (25), (26) and (27), we get that Eq. (24) also holds.

Further, the map ψ defined by Eq. (18) is single-valued and continuous (this follows from the fact that φ is a single-valued and continuous map). Let us show that the map ψ is M-fejerian. Let $x \in \mathbb{R}^n \backslash M$ be an arbitrary point not belonging to the polytope M. Let us choose a point $\hat{x} \in M$ that satisfies the following condition

$$\left\| \varphi^L(x) - \hat{x} \right\| = \operatorname{dist}(\varphi^L(x), M). \tag{29}$$

Such a point exists and is unique because the polytope M is a bounded, closed and convex set. Combining the *dist* definition, Eq. (18), the triangle inequality for the norm and Eqs. (14) and (29), we get

$$\begin{aligned} &\operatorname{dist}(\psi(x), M) \le \|\psi(x) - \hat{x}\| = \left\| \varphi^L(x) - Ld - \hat{x} \right\| \\ &\le \|Ld\| + \left\| \varphi^L(x) - \hat{x} \right\| = \|Ld\| + \operatorname{dist}(\varphi^L(x), M) < \operatorname{dist}(x, M). \end{aligned}$$

It follows that ψ is M-fejerian. Therefore,

$$\left\{ \psi^k(y_0) \right\}_{k=0}^{+\infty} \to \bar{y} \in M.$$

This means that $\lim\limits_{k \to \infty} \operatorname{dist}(y_k, M) = 0$. Taking into account (24), we conclude that $\lim\limits_{k \to \infty} \operatorname{dist}(z_k, M_k) = 0$. This completes the proof of the theorem.

From a non-formal point of view, Theorem 1 states that the Fejer process must converge faster than the polytope M "runs away". Manycore processors can be used to increase the Fejer mapping calculation speed. In [19], the authors investigated this issue on Intel Xeon Phi multi-core coprocessors with MIC architecture [21]. It was shown that the Intel Xeon Phi may be used efficiently for solving large-scale problems.

3.2 The *Targeting* Phase

The *Targeting* phase begins after the *Quest* phase. At the *Targeting* phase, an n-dimensional axisymmetric cross is formed. An *n-dimensional axisymmetric cross* is a finite set $G = \{g_0, \ldots, g_P\} \subset \mathbb{R}^n$. The cardinality of G equals $P + 1$, where P is a multiple of $n \ge 2$. The point g_0 is singled out from the point set G. This point is called the *central point* of the cross. Initially, the central point is assigned the coordinates of the point z_k calculated in the *Quest* phase by using the iterative process (10).

The set $G \backslash \{g_0\}$ is divided into n disjoint subsets C_i $(i = 0, \ldots, n-1)$ called the *cohorts*:

$$G \backslash \{g_0\} = \bigcup_{i=0}^{n-1} C_i,$$

where n is the dimension of the space. Each cohort C_i consists of

$$K = \frac{P}{n} \tag{30}$$

points lying on the straight line that is parallel to the i-th coordinate axis and passes through the central point g_0. By itself, the central point does not belong to any cohort. The distance between any two neighbor points of the set $C_i \cup \{g_0\}$ is equal to a constant s. An example of a two-dimensional cross is shown in Fig. 6. The number of points in one dimension, excluding the central point, is equal to K. The symmetry of the cross supposes that K takes only even values greater than or equal to 2. Using Eq. (30), we obtain the following equation for the total number of points contained in the cross:

$$P + 1 = nK + 1. \tag{31}$$

Since K can take only even values greater than or equal to 2 and $n \geq 2$, it follows from Eq. (31) that P can also take only even values, and $P \geq 4$. In Fig. 6, we have $n = 2$, $K = 6$, $P = 12$.

Each point of the cross G is uniquely identified by a marker being a pair of integer numbers (χ, η) such that $0 \leq \chi < n$, $|\eta| \leq K/2$. Informally, χ specifies the number of the cohort, and η specifies the sequential number of the point in the cohort C_χ counted from the central point. The corresponding marking of points in the two-dimensional case is given in Fig. 6(a). The coordinates of the point $x_{(\chi,\eta)}$ having the marker (χ, η) can be reconstructed as follows:

$$x_{(\chi,\eta)} = g_0 + (0, \ldots, 0, \underbrace{\eta \cdot s}_{\chi}, 0, \ldots, 0). \tag{32}$$

The vector being added to g_0 in the right part of Eq. (32) has a single non-zero coordinate in position χ. This coordinate equals $\eta \cdot s$, where s is the distance between neighbor points in a cohort.

The *Targeting* phase includes the following steps.

1. Build the n-dimensional axisymmetric cross G that has K points in each cohort, the distance between neighbor points equaling s, and the center at point $g_0 = z_k$, where z_k is obtained in the *Quest* phase.
2. Calculate $G' = G \cap M_k$.
3. Calculate $C'_\chi = C_\chi \cap G'$ for $\chi = 0, \ldots, n-1$.
4. Calculate $Q = \bigcup_{\chi=0}^{n-1} \{\arg\max \{\langle c_k, g\rangle \mid g \in C'_\chi, C'_\chi \neq \emptyset\}\}$.
5. If $g_0 \in M_k$ and $\langle c_k, g_0\rangle \geq \max\limits_{q \in Q} \langle c_k, q\rangle$, then $k := k + 1$, and go to step 2.
6. $g_0 := \frac{\sum_{q \in Q} q}{|Q|}$.
7. $k := k + 1$.
8. Go to step 2.

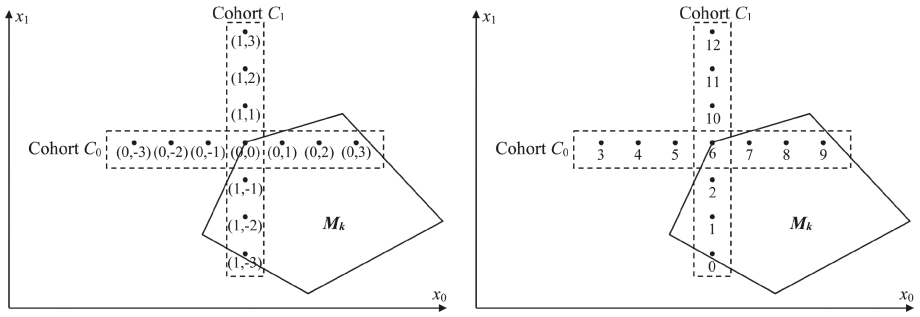

Fig. 6. A two-dimensional cross

Thus, in the *Targeting* phase, the steps 2–7 form a perpetual loop in which the approximate solution of the non-stationary LP problem is permanently recalculated. From a non-formal point of view, in Step 2, we determine which points of the cross G belong to the polytope M_k. To do this, we check the condition $A_k g \le b_k$ for each point $g \in G$. Such checks can be executed in parallel by different processor nodes of a cluster computing system. For this goal to be achieved, P MPI-processes can be exploited, where P is defined by Eq. (31). We use sequential numbering for distributing the cross points among the MPI-processes. Each point of the cross is assigned a unique number $\alpha \in \{0, \ldots, P-1\}$. The sequential number α can be converted to a marker (χ, η) by means of the following equations[3]:

$$\chi = ||\alpha - K| - 1| \div (K/2);$$
$$\eta = \operatorname{sgn}(\alpha - K) \cdot (((|\alpha - K| - 1) \bmod (K/2)) + 1).$$

The backward conversion can be performed by means of the equation

$$\alpha = \eta + \operatorname{sgn}(\eta)\frac{\chi}{2}K + K.$$

Figure 6(b) demonstrates the sequential numbering of points that corresponds to the marking in Fig. 6(a).

4 Conclusion

In this paper, a new *NSLP* algorithm for solving non-stationary linear programming problems of large dimension has been described. This algorithm is oriented to cluster computing systems with manycore processors. The algorithm consists of two phases: *Quest* and *Targeting*. The *Quest* phase calculates a solution of the system of inequalities defining the constraint system of the linear programming

[3] The symbol $\div$ denotes integer division.

problem under the condition of input data dynamic changes. To do this, we organize a Fejer process that computes a pseudo-projection onto the polytope M defined by the constraints of the LP problem. In this case, input data changes occur during calculation of the pseudo-projection. A convergence theorem for the described iterative process is proved in the case of translation of the polytope M. The *Targeting* phase forms a special system of points having the shape of an n-dimensional axisymmetric cross. The cross moves in the n-dimensional space in such a way that the solution of the linear programming problem is located all the time in an ε-vicinity of the central point of the cross. A formal description of the *Targeting* phase is presented in the form of a sequence of steps. Our future goal is a parallel implementation of the *NSLP* algorithm in the C++ language using the MPI library, as well as the development of computational experiments on a cluster computing system using synthetic and real LP problems.

References

1. Chung, W.: Applying large-scale linear programming in business analytics. In: Proceedings of the 2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), pp. 1860–1864. IEEE (2015)
2. Tipi, H.: Solving super-size problems with optimization. Presentation at the meeting of the 2010 INFORMS Conference on O.R. Practice. Orlando, Florida, April 2010. http://nymetro.chapter.informs.org/prac_cor_pubs/06-10%20Horia%20Tipi%20SolvingLargeScaleXpress.pdf. Accessed 7 May 2017
3. Gondzio, J., et al.: Solving large-scale optimization problems related to Bells Theorem. J. Comput. Appl. Math. **263**, 392–404 (2014)
4. Sodhi, M.S.: LP modeling for asset-liability management: a survey of choices and simplifications. Oper. Res. **53**(2), 181–196 (2005)
5. Dyshaev, M.M., Sokolinskaya, I.M.: Predstavlenie torgovykh signalov na osnove adaptivnoy skol'zyashchey sredney Kaufmana v vide sistemy lineynykh neravenstv [Representation of trading signals based Kaufman adaptive moving average as a system of linear inequalities]. Vestnik Yuzhno-Ural'skogo gosudarstvennogo universiteta. Seriya: Vychislitel'naya matematika i informatika [Bull. South Ural State Univ. Ser. Comput. Math. Softw. Eng.] **2**(4), 103–108 (2013)
6. Ananchenko, I.V., Musaev, A.A.: Torgovye roboty i upravlenie v khaoticheskikh sredakh: obzor i kriticheskiy analiz [Trading robots and management in chaotic environments: an overview and critical analysis]. In: Trudy SPIIRAN [SPIIRAS Proceedings], vol. 3, no. 34, pp. 178–203 (2014)
7. Radenkov, S.P., Gavryushin, S.S., Sokolyanskiy, V.V.: Avtomatizirovannyye torgovyye sistemy i ikh installyatsiya v rynochnuyu sredu (chast' 1) [Automated trading systems and their installation in the market environment (Part 1)]. Voprosy ekonomicheskikh nauk [Probl. Econ.] **6**(76), 70–74 (2015)
8. Dantzig, G.: Linear Programming and Extensions. Princeton University Press, Princeton (1998). 656 pp.
9. Klee, V., Minty, G.J.: How good is the simplex algorithm? In: Proceedings of the Third Symposium on Inequalities, University of California, Los Angeles, CA, pp. 159–175. Academic Press, New York-London, 1–9 September 1969. Dedicated to the Memory of Theodore S. Motzkin

10. Khachiyan, L.G.: Polynomial algorithms in linear programming. USSR Comput. Math. Math. Phys. **20**(1), 53–72 (1980)
11. Shor, N.Z.: Cut-off method with space extension in convex programming problems. Cybern. Syst. Anal. **13**(1), 94–96 (1977)
12. Karmarkar, N.: A new polynomial-time algorithm for linear programming. In: Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing, pp. 302–311. ACM (1984)
13. Sokolinskaya, I.M., Sokolinskii, L.B.: Parallel algorithm for solving linear programming problem under conditions of incomplete data. Autom. Remote Control **71**(7), 1452–1460 (2010)
14. Rechkalov, T.V., Zymbler, M.L.: Accelerating medoids-based clustering with the Intel many integrated core architecture. In: Proceedings of the 9th International Conference on Application of Information and Communication Technologies, Rostov-on-Don, Russia, pp. 413–417. IEEE, 14–16 October 2015
15. Zymbler, M.: Best-match time series subsequence search on the intel many integrated core architecture. In: Morzy, T., Valduriez, P., Bellatreche, L. (eds.) ADBIS 2015. LNCS, vol. 9282, pp. 275–286. Springer, Cham (2015). doi:10.1007/978-3-319-23135-8_19
16. Sokolinskaya, I.M., Sokolinsky, L.B.: Implementation of parallel pursuit algorithm for solving unstable linear programming problems. Bull. South Ural State Univ. Ser. Comput. Math. Softw. Eng. **5**(2), 15–29 (2016). doi:10.14529/cmse160202. (in Russian)
17. Sokolinskaya, I., Sokolinsky, L.: Solving unstable linear programming problems of high dimension on cluster computing systems. In: Proceedings of the 1st Russian Conference on Supercomputing - Supercomputing Days 2015, Moscow, Russian Federation. CEUR Workshop Proceedings, vol. 1482, pp. 420–427. CEUR-WS.org, 28–29 September 2015
18. Eremin, I.I.: Fejerovskie metody dlya zadach linejnoj i vypukloj optimizatsii [Fejer Methods for Problems of Convex and Linear Optimization]. Publishing of the South Ural State University, Chelyabinsk (2009). 200 pp.
19. Sokolinskaya, I., Sokolinsky, L.: Revised pursuit algorithm for solving non-stationary linear programming problems on modern computing clusters with many-core accelerators. In: Voevodin, V., Sobolev, S. (eds.) RuSCDays 2016. CCIS, vol. 687, pp. 212–223. Springer, Cham (2016). doi:10.1007/978-3-319-55669-7_17
20. Eremin, I.I.: Teoriya lineynoy optimizatsii [The theory of linear optimization]. Publishing House of the "Yekaterinburg", Ekaterinburg (1999). 312 pp.
21. Thiagarajan, S.U., Congdon, C., Naik, S., Nguyen, L.Q.: Intel Xeon Phi coprocessor developers quick start guide. White Paper. Intel (2013). https://software.intel.com/sites/default/files/managed/ee/4e/intel-xeon-phi-coprocessor-quick-start-developers-guide.pdf. Accessed 7 May 2017