Scalable algorithm for non-stationary linear programming problems solving

Irina M. Sokolinskaya
School of Electrical Engineering and Computer Science
South Ural State University (national research university)
Chelyabinsk, Russian Federation
Irina.Sokolinskaya@susu.ru

Abstract—**The paper describes a new scalable algorithm called NSLP for solving high-dimension, non-stationary linear programming problem on the 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 condition of the dynamic changes of input data. To do this, it uses the apparatus of Fejer maps. The Targeting phase forms a special system of points having the shape of the n-dimensional axisymmetric cross. The cross moves in the n-dimensional space in such a way that the solution of the linear programming problem permanently was in the ε-vicinity of the central point of the cross.**

Keywords—NSLP-algorithm, non-stationary linear programming problem, large-scale linear programming, Fejer map

I. Introduction

New measurement technology in the industry has spawned the large-scale linear programming (LP) problems. Each of these problems uses Big Data from subject area. Such problem is formalized as a linear programming problem that involve up to tens of millions of constraints and up to hundreds of millions of decision variables. In many cases, the LP problems arising in the industry are nonstationary (dynamic). The period of input data change can be within the range of hundredths of a second.

Until now, one of the most popular methods solving LP problems is the class of algorithms proposed and designed by Dantzig on the base of the simplex method [1]. The simplex method has proved to be effective in solving a large class of LP problems. However, Klee and Minty [2] gave an example showing that the worst-case complexity of simplex method is exponential time. Nevertheless, Khaciyan [3] proved that LP problem can be solved in polynomial time by a variant of an iterative ellipsoidal algorithm developed by Shor [4]. Attempts to apply the ellipsoidal algorithm in practice were unsuccessful, since, in most cases, this algorithm demonstrated much worse efficiency than the simplex did. Karmarkar in [5] proposed a method for linear programming called "Interior point method" which runs in polynomial time and is also very efficient in practice.

The simplex method and the method of interior points 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 author in [6] described a parallel algorithm for solving LP problems with not-formalized constraints. The main idea of the proposed approach is to combine linear programming and discriminant analysis methods. The discriminant analysis demands two sets of patterns M and N. The first set must satisfy the not-formalized constraints and the second mustn't. To obtain representative patterns, the methods of data mining [7] and time series analysis can be used [8]. To overcome the problem of non-stationarity of input data, the authors proposed in [9], [10] the Pursuit algorithm for solving non-stationary LP problems on cluster computing systems. The Pursuit algorithm uses Fejer maps [11] to build a pseudo-projection onto the convex bounded set. The pseudo-projection operator is like the projection, but, in contrast to the last, the pseudo-projection is stable for dynamic change of input data. In the paper [12], the author investigated the efficiency of using multi-core processors Intel Xeon Phi to calculate the pseudo-projections.

This paper describes the new algorithm NSLP (Non-Stationary Linear Programming) 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 once (the Pursuit algorithm computes pseudo-projections K times at each iteration, where K is the number of processor nodes). The rest of the paper is organized as follows. Section II gives a formal statement of a LP problem and presents the definitions of Fejer process and the pseudo-projection onto a polyhedron. Section III describes the new NSLP algorithm. Section IV summarizes the results obtained and proposes the directions for future research.

II. Problem Statement

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

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

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 the time $t \in \mathbb{R}_{\ge 0}$. We assume that the value of $t = 0$ corresponds to the initial instant of time:

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

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

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

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

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

Let M_t be a polyhedron defined by the constraints of the non-stationary LP problem (1). Such a polyhedron is always convex. It's known [[11]] that φ_t will be a continuous single-valued M_t-fejerian map for the relaxation factor $0 < \lambda < 2$.

By definition, put

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

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 [[13]]) that Fejer process for fixed t converges to a point belonging to the polyhedron M:

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

Let us consider the simplest case of non-stationarity, which is a translation of the polyhedron $M = M_0$ 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-stationarity problem (1) takes the following form:

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

what is equivalent to

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

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

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

what is equivalent to

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

Let φ-*projection* (*pseudo-projection*) of point $x \in \mathbb{R}^n$ on polyhedron M be understood as the map $\pi_M^{\varphi}(x) = \lim_{s\to\infty}\varphi^s(x)$.

III. *NSLP* ALGORITHM

The *NSLP* (*Non-Stationary Linear Programming*) algorithm is designed to solve large-scale non-stationary LP problems on cluster computing systems. It includes 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 condition of the dynamic changes of input data. To do this, it uses the apparatus of Fejer maps. The *Targeting* phase forms a special system of points having the shape of the *n*-dimensional axisymmetric cross. The cross moves in the *n*-dimensional space in such a way that the solution of the LP problem permanently was in the *ε*-vicinity of the central point of the cross. Let us describe both phases of the algorithm in more detail.

A. 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 $z_0 \in \mathbb{R}_{\ge 0}^n$ with non-negative coordinates. This point plays the role of initial approximation of the problem (1). Then we organize the iterative Fejer process of the form (6), during which the Fejer approximations consecutively are calculated by using the Fejer mapping (3). This process converges to a point located on the polyhedron M_t. In according to the non-stationary nature of the problem (1), the polyhedron M_t can change its position and shape during calculating the pseudo-projection. Every *L* iterations, the input data update is performed, where *L* is some fixed positive integer that is a parameter of the algorithm. Let us denote by $t_0, t_1, \ldots, t_k, \ldots$ the sequential points of time corresponding to instants of the input data update. Without loss of generality, we can assume that

$$t_0 = 0, t_1 = L, t_2 = 2L, \ldots, t_k = kL, \ldots \quad (9)$$

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

Let the polyhedron M_t take shapes and locations $M_0, M_1, \ldots, M_k, \ldots$ at the points of time (9). Let $\varphi_0, \varphi_1, \ldots, \varphi_k, \ldots$ be Fejer maps determined by formula (3), taking into account the changes of the input data of problem (1) at the points of time (9). In the *Quest* phase, the iterative process calculates the following sequence of points:

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

Let us briefly denote this iterative process as follows:

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

This iterative process ends when

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

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

Theorem 1. Let non-stationary LP problem be given by (7). Let Fejer maps $\varphi_0, \varphi_1, \ldots, \varphi_k, \ldots$ be given by the following formula:

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

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

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

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

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

under the following condition

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

The theorem 1 gives a sufficient condition for the convergence of the iterative process for the non-stationary problem (7). In order to prove this theorem, we will need the following auxiliary lemma.

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

$$v - u = pLd \Rightarrow \varphi_p^l(v) - \varphi^l(u) = pLd \quad (15)$$

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

Proof. The proof is by induction on l. Induction basis: let $l = 1$ and the following condition holds

$$v - u = pLd. \quad (16)$$

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

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

Therefore, (15) holds if $l = 1$. Inductive step: assume that condition (16) is true. Using the induction hypothesis, we get

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

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

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

This completes the proof □

Proof of Theorem 1. Let us fixe an arbitrary point $z_0 \in \mathbb{R}^n \setminus M$. Let the map $\psi : \mathbb{R}^n \to \mathbb{R}^n$ be given by:

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

By definition, put

$$y_0 = z_0 \quad (19)$$

and

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

for $k = 1, 2, \ldots$.

Using induction on k, let us show that

$$z_k - y_k = kLd \quad (21)$$

for $k = 0, 1, 2, \ldots$. Induction basis: the equation (21) holds for $k = 0$, since, taking into account (19), the equation

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

holds.

Inductive step: suppose

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

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

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

Comparing this with (22), we have

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

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

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

Therefore, equation (21) holds.

Now we show that

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

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

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

Such a point exists and is unique, since the polyhedron M is a bounded, closed, convex set. The polyhedron M_k is a result of translating the polyhedron M by the vector kLd. Since $\hat{y} \in M$, it follows that the point $\hat{z} = \hat{y} + kLd$ belongs to the polyhedron 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. Hence,

$$\|\hat{z} - z_k\| = \|\hat{y} - y_k\|\ . \qquad (26)$$

Let us show that

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

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

$$\|z' - z_k\| < \|\hat{z} - z_k\|\ . \qquad (28)$$

Since $z' \in M_k$, it follows that the point $y' = z' - kLd$ belongs to the polyhedron 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 (28), (26) and (25), 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. Hence, the formula (27) holds. Combining (25), (26) and (27), we get that the formula (24) holds.

Further, the map ψ defined by the formula (18) is single-valued and continuous (it follows from the single-valuedness and continuity of the map φ). Let us show that the map ψ is *M*-fejerian. Let $x \in \mathbb{R}^n \setminus M$ be an arbitrary point not belonging to the polyhedron M. Let us choose a point $\hat{x} \in M$ that satisfies the following condition

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

Such a point exists and is unique, since the polyhedron M is a bounded, closed, convex set. Combining the *dist* definition, the formula (18), the triangle inequality for the norm, the formulas (29) and (14), we get

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

It follows that ψ is *M*-fejerian. Hence,

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

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

From the non-formal point of view, Theorem 1 says that the Fejer process must converge faster than the polyhedron M "runs away". To increase the speed of Fejer map calculation, manycore processors can be used. In the paper [[12]], this issue was investigated on multi-core coprocessors of Intel Xeon Phi with MIC architecture [[14]]. It was shown that the use of Intel Xeon Phi is efficient for solving large-scale problems.

B. Targeting phase

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

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

$$G \setminus \{g_0\} = \bigcup_{i=0}^{n-1} C_i\ .$$

Each *i*-th cohort ($i = 0, \ldots, n-1$) consists of

$$K = P/n \qquad (30)$$

points lying on the straight line, which is parallel to the *i*-th coordinate axis and passing 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 $G \cup \{g_0\}$ is equal to the constant s. 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 formula (30), we obtain the following formula giving the total number of points in the cross:

$$P + 1 = nK + 1. \quad (31)$$

Since K can take only even values greater than or equal to 2 and $n \geq 2$, from formula (31), it follows that P can also take only even values and $P \geq 4$.

Each point of the cross G is uniquely identified by a *marker* being a pair of integers 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 sequence number of the point in the cohort C_χ, being counted out of the central point. The coordinates of the point $x_{(\chi,\eta)}$ having the marker (χ, η) can be reconstructed as follows:

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

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

Targeting phase includes the following steps.

1. Build the *n*-dimensional axisymmetric cross G, which has K points in a cohort, the distance between neighbor points equaling s, and the center at point $g_0 = z_k$, where z_k is obtained as a result of 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 $Q = \bigcup_{\chi=0}^{n-1} \left\{ \arg\max \left\{ \langle c_k, g \rangle \middle| g \in C'_\chi, C'_\chi \neq \varnothing \right\} \right\}$.
5. If $g_0 \in M_k$ and $\langle c_k, g_0 \rangle \geq \max_{q \in Q} \langle c_k, q \rangle$ then $k := k+1$ and go to the step 2.
6. $g_0 := \dfrac{\sum_{q \in Q} q}{|Q|}$.
7. $k := k+1$.
8. Go to the step 2.

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 the non-formal point of view, in Step 2, we determine which points of the cross G are belonged to the polyhedron M_k. To do it, we check the condition $A_k g \leq b_k$ for every point $g \in G$. Such checks can be executed in parallel by the different processor nodes of a cluster computing system. For it, *P* of MPI-processes can be exploited, where *P* is defined by the formula (31). To distribute the points among the MPI-processes, we use the sequential numbering. Each point of the cross G is assigned a unique number $\alpha \in \{0,\ldots,P-1\}$. The sequential number α can be converted to a marker (χ, η) by the following formulas (symbol $\div$ denotes the integer division):

$$\chi = \left\| \alpha - K \right| - 1 \right| \div (K/2); \quad (33)$$

$$\eta = \operatorname{sign}(\alpha - K) \cdot \left(\left(\left(|\alpha - K| - 1 \right) \bmod (K/2) \right) + 1 \right). \quad (34)$$

The backward conversion can be performed by the formula

$$\alpha = \eta + \operatorname{sign}(\eta) \frac{\chi}{2} K + K. \quad (35)$$

IV. Conclusion

In this paper, a new NSLP algorithm for solving non-stationary linear programming problems of large-scale 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 problem under condition of the dynamic changes of input data. To do this, we organize Fejer process computing a pseudo-projection onto the polyhedron M defined by the constraints of the LP problem. In this case, the input data changes during the calculation of the pseudo-projection. For the described iterative process, the convergence theorem is proved in case of translating the polyhedron M. The *Targeting* phase forms a special system of points having the shape of the *n*-dimensional axisymmetric cross. The cross moves in the *n*-dimensional space in such a way that the solution of the linear programming problem permanently was in the 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 C++ language using MPI library, and computational experiments on a cluster computing system using synthetic and real LP problems.

References

[1] Dantzig G. Linear programming and extensions. Princeton, N.J.: Princeton university press, 1998. 656 pp.

[2] Klee V., Minty G.J. How good is the simplex algorithm? // Inequalities III (Proceedings of the Third Symposium on Inequalities held at the University of California, Los Angeles, Calif., September 1–9, 1969, dedicated to the memory of Theodore S. Motzkin). New York-London: Academic Press, 1972. P. 159–175.

[3] Khachiyan L. G. Polynomial algorithms in linear programming // USSR Computational Mathematics and Mathematical Physics. 1980. T. 20. No. 1. P. 53-72.

[4] Shor N.Z. Cut-off method with space extension in convex programming problems // Cybernetics and Systems Analysis. 1977. Vol. 13, № 1. P. 94–96.

[5] Karmarkar N. A new polynomial-time algorithm for linear programming // Proceedings of the sixteenth annual ACM symposium on Theory of computing. ACM, 1984. P. 302-311.

[6] Sokolinskaya I.M., Sokolinskii L.B. Parallel algorithm for solving linear programming problem under conditions of incomplete data // Automation and Remote Control. 2010. Vol. 71, No. 7. P. 1452-1460.

[7] Rechkalov T.V., Zymbler M.L. Accelerating Medoids-based Clustering with the Intel Many Integrated Core Architecture // Proceedings of the 9th International Conference on Application of Information and Communication Technologies (AICT'2015), October 14–16, 2015, Rostov-on-Don, Russia. IEEE, 2015. P. 413–417.

[8] Zymbler M.L. Best-match Time Series Subsequence Search on the Intel Many Integrated Core Architecture // Proceedings of the 19th East-European Conference on Advances in Databases and Information Systems, ADBIS 2015 (Poitiers, France, September 8–11, 2015). Lecture Notes in Computer Science. Vol. 9282. Springer, 2015. P. 275–286.

[9] Sokolinskaya I.M., Sokolinsky L.B. Implementation of Parallel Pursuit Algorithm for Solving Unstable Linear Programming Problems. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2016. vol. 5, no. 2. pp. 15–29. (in Russian) DOI: 10.14529/cmse160202.

[10] Sokolinskaya I., Sokolinsky L. Solving unstable linear programming problems of high dimension on cluster computing systems // 1st Russian Conference on Supercomputing Days 2015, RuSCDays 2015; Moscow; Russian Federation; 28 September 2015 through 29 September 2015. CEUR Workshop Proceedings. V. 1482, CEUR-WS.org 2015. P. 420-427.

[11] Eremin I.I. Fejerovskie metody dlya zadach linejnoj i vypukloj optimizatsii [Fejer's Methods for Problems of Convex and Linear Optimization]. Chelyabinsk, Publishing of the South Ural State University, 2009. 200 p.

[12] Sokolinskaya I., Sokolinsky L. B. Revised Pursuit Algorithm for Solving Non-Stationary Linear Programming Problems on Modern Computing Clusters with Manycore Accelerators // Supercomputing. RuSCDays 2016. Communications in Computer and Information Science. 2016. Vol. 687. P. 212-223. DOI: 10.1007/978-3-319-55669-7_17.

[13] Eremin I.I. Teoriya lineynoy optimizatsii [The theory of linear optimization]. Ekaterinburg: Publishing House of the "Yekaterinburg", 1999. 312 pp.

[14] Thiagarajan S.U., Congdon C., Naik S., Nguyen L.Q. Intel Xeon Phi coprocessor developer's quick start guide. White Paper. Intel, 2013. URL: https://software.intel.com/sites/default/files/managed/ee/4e/intel-xeon-phi-coprocessor-quick-start-developers-guide.pdf (accessed 04.03.2017).