Решение нестационарных задач линейного программирования большой размерности на кластерных вычислительных системах

И.М. Соколинская, Л.Б. Соколинский

Южно-Уральский государственный университет

В работе описывается параллельный алгоритм решения нестационарных задач линейного программирования большой размерности, ориентированный на кластерные вычислительные системы. В основе алгоритма, получившего название «следящий», лежат фейеровские отображения. Алгоритм отслеживает изменения исходных данных и вносит корректировки в вычислительный процесс. При этом задача разбивается на большое количество подзадач, которые могут решаться независимо без обменов данными. Приводятся диаграммы деятельности UML, описывающие параллельный следящий алгоритм.

1. Введение

Одной из особенностей современного экономико-математического моделирования является появление нестационарных задач линейного программирования [1] большой размерности с быстро меняющимися входными данными. Кроме этого, часть ограничений может оказаться плохо формализуемой [2]. Одним из примеров таких задач является задача управления портфелем ценных бумаг с использованием методов алгоритмической торговли [3, 4]. В подобных задачах количество переменных и неравенств в системе ограничений может составлять десятки и даже сотни тысяч, а период изменения исходных данных находиться в пределах сотых долей секунды. В работе [5] был описан параллельный алгоритм для решения задач линейного программирования с плохо формализуемыми ограничениями. Для преодоления проблемы нестационарности входных данных в работах [6, 7] был предложен *«следящий»* алгоритм решения задачи линейного программирования с использованием фейеровских отображений [8], ориентированный на кластерные вычислительные системы с многоядерными ускорителями. В данной работе дается параллельная реализация следящего алгоритма с использованием диаграммы деятельности UML. Статья организована следующим образом. В разделе 2 приводится формальная постановка задачи линейного программирования, даются определения фейеровского процесса и операции псевдопроектирования на многогранник. В разделе 3 приводится неформальное описание следящего алгоритма. В разделе 4 описывается межузловое распараллеливание следящего алгоритма, предполагающее использование технологии параллельного программирования MPI. В заключении суммируются полученные результаты и определяются направления дальнейших исследований.

2. Формализация задачи

Пусть задана задача линейного программирования

$$\max\left\{\langle c,x\rangle \,|Ax\le b,\ x\ge 0\right\}. \tag{1}$$

Определим отображение $\varphi:\mathbb{R}^n\to\mathbb{R}^n$ следующим образом:

$$\varphi(x)=x-\sum_{j=1}^{m}\alpha_j\lambda_j\frac{\max\left\{\langle a_j,x\rangle-b_j,0\right\}}{\|a_j\|^2}\cdot a_j. \tag{2}$$

Пусть M – многогранник, задаваемый ограничениями задачи линейного программирования (1). Такой многогранник всегда является выпуклым. Известно [8], что φ будет однозначным непрерывным M-фейеровским отображением для любой системы положительных коэффициентов $\{\alpha_j > 0\}$ $(j = 1, \ldots, m)$, таких, что $\sum\limits_{j=1}^{m} \alpha_j = 1$, и коэффициентов релаксации $0 < \lambda_j < 2$. Полагая в формуле (2) $\lambda_j = \lambda$ и $\alpha_j = 1/m$ $(j = 1, \ldots, m)$, получаем формулу

$$\varphi(x) = x - \frac{\lambda}{m} \sum_{j=1}^{m} \frac{\max\{\langle a_j, x\rangle - b_j, 0\}}{\|a_j\|^2} \cdot a_j, \tag{3}$$

которая будет использоваться в следящем алгоритме.

Обозначим

$$\varphi^s(x) = \underbrace{\varphi \ldots \varphi}_{s}(x).$$

Под *фейеровским процессом*, порождаемым отображением φ при произвольном начальном приближении $x_0 \in \mathbb{R}^n$, будем понимать последовательность $\{\varphi^s(x_0)\}_{s=0}^{+\infty}$. Известно, что указанный фейеровский процесс сходится к точке, принадлежащей множеству M:

$$\{\varphi^s(x_0)\}_{s=0}^{+\infty} \to \bar{x} \in M.$$

Будем кратко обозначать это следующим образом: $\lim\limits_{s\to\infty} \varphi^s(x_0) = \bar{x}$.

Под *φ-проектированием* (*псевдопроектированием*) точки $x \in \mathbb{R}^n$ на многогранник M понимается отображение $\pi_M^{\varphi} : \mathbb{R}^n \to M$ [9], задаваемое соотношением $\pi_M^{\varphi}(x) = \lim\limits_{s\to\infty} \varphi^s(x)$.

3. Следящий алгоритм

В общем виде следящий алгоритм может быть описан следующим образом. Все пространство $\mathbb{R}^n$ делится гиперкубической сеткой на ячейки с переменной длиной грани. Размер ячейки может динамически меняться в зависимости от скорости изменения исходных данных: чем быстрее меняются данные, тем больше становится ячейка, чтобы успевать следить. Алгоритм постоянно находит (отслеживает) ячейку наименьшего размера, на которой достигается максимум целевой функции. Будем такую ячейку называть целевой. В каждой итерации алгоритма просчитываются ячейки, входящие в некоторую следящую область вокруг целевой ячейки. В простейшем случае следящая область может быть кубической формы со следящей ячейкой в центре. В общем случае следящая область может быть произвольной выпуклой фигурой. При этом ячейки всегда имеют кубическую форму. *Целевая точка* – это некоторая точка, находящаяся вне следящей области. Ее координаты могут динамически вычисляться по коэффициентам целевой функции. Например, при целевой функции $Q(x) = x_1 + 2x_2$ в качестве целевой точки можно взять точку $(T, 2T)$, где T – достаточно большое положительное число. *Нулевой вершиной* кубической области будем называть вершину куба, находящуюся ближе всего к началу координат. *Нулевой вершиной* кубической ячейки будем называть вершину ячейки, находящуюся ближе всего к началу координат.

Неформально следящий алгоритм может быть описан следующей последовательностью действий.

1. Первоначально выбирается следящая кубическая область с длиной ребра r и нулевой вершиной в точке q, заведомо покрывающая многогранник.
2. Выбирается целевая точка $z = Tc$, расположенная вне следящей области.
3. Следящая область разбивается на K ячеек.

4. В условиях динамического изменения входных данных (A, b, c), для всех ячеек внутри следящей области вычисляется псевдопроекция из точки z на пересечение ячейки с многогранником. Если пересечение пусто, то такие ячейки отбрасываются.

5. Если получено пустое множество псевдопроекций, то мы увеличиваем размер следящей области в w раз и переходим на шаг 2.

6. Если получено непустое множество псевдопроекций, то на нем вычисляется максимум целевой функции.

7. Если расстояние от точки максимума до центра следящей области меньше $\frac{1}{4}r$, то длина ребра следящей области r уменьшается в 2 раза.

8. Если расстояние от точки максимума до центра следящей области больше $\frac{3}{4}r$, то длина ребра следящей области r увеличивается в 2 раза.

9. Центр следящей области перемещается в центр ячейки, в которой достигнут максимум, и переходим на шаг 2.

Псевдопроекции на шаге 4 для различных ячеек следящей области могут вычисляться параллельно без обменов данными между MPI-процессами. Фейеровский процесс вычисления псевдопроекции, стартовавший из целевой точки на шаге 4, вычисляется до конца, несмотря на то, что координаты целевой точки и сам многогранник в процессе счета могли измениться. Выполнение условия шага 5 означает, что входные данные изменяются быстрее, чем мы вычисляем псевдопроекции. Константы 1/2 и 3/4 , используемые при выполнении шагов 7 и 8, являются значениями параметров алгоритма.

4. Межузловое распараллеливание

Межузловое распараллеливание – это распараллеливание вычислений между отдельными узлами кластерной вычислительной системы. В качестве технологии параллельного программирования используется MPI. Каждый MPI-процесс работает на отдельном узле кластера. Обмен данными между MPI-процессами осуществляется с помощью передачи сообщений по коммуникационной сети. Один MPI-процесс считает одну псевдопроекцию на пересечение одной ячейки следящей области с многогранником. Таким образом, количество ячеек следящей области всегда равно константе K, совпадающей с количеством MPI-процессов.

Опишем метод межузлового распараллеливания следящего алгоритма. Без ограничения общности мы можем считать, что все процессы происходят в положительной области координат. Пусть P – количество доступных MPI-процессов (количество процессорных узлов в многопроцессорной системе). Определим общее количество ячеек в следящей области следующим образом:

$$K = \left(\left\lfloor P^{1/n} \right\rfloor \right)^n. \tag{4}$$

Тогда количество ячеек, примыкающих к одному ребру куба следящей области, равно $h = K^{1/n}$. Зададим в пространстве целочисленных координат $u_0, \ldots, u_n$ линейную нумерацию ячеек следящей области следующим образом. Пусть ячейка α имеет целочисленные координаты $(\alpha_0, \ldots, \alpha_n)$. Тогда ее номер k_α вычисляется по формуле:

$$k_\alpha = \sum_{i=0}^{n-1} \alpha_i n^i. \tag{5}$$

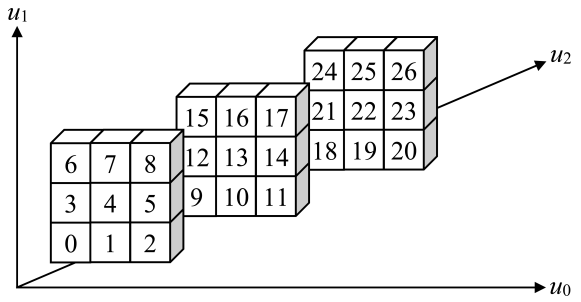

Рис. 1. Линейная нумерация ячеек следящей области при $n = 3$.

На рис. 1 приведен пример такой линейной нумерации при $n = 3$. Например, ячейка с номером 19 на рис. 1 имеет целочисленные координаты $(1, 0, 2)$. Действительно, $19 = 1 \cdot 3^0 + 0 \cdot 3^1 + 2 \cdot 3^2$.

Выразим целочисленные координаты ячейки α через ее порядковый номер k_α. Из (5) получаем

$$\alpha_0 = k_\alpha \bmod n; \tag{6}$$

$$\alpha_1 = \frac{k_\alpha - \alpha_0}{n} \bmod n; \tag{7}$$

$$\alpha_2 = \frac{k_\alpha - \alpha_0 - \alpha_1 n}{n^2} \bmod n; \tag{8}$$

$$\cdots\cdots\cdots$$

Таким образом, в общем виде имеем:

$$\alpha_i = \frac{k_\alpha - \sum\limits_{j=0}^{i-1} \alpha_j n^j}{n^i} \bmod n. \tag{9}$$

Формула (9) содержит ресурсоемкую операцию возведения в степень. От нее можно избавиться следующим образом. По определению операции mod из (6) получаем

$$\alpha_0 = k_\alpha - (k_\alpha \div n) \cdot n. \tag{10}$$

С помощью символа $\div$ здесь обозначается целочисленное деление. Подставив в (7) вместо α_0 правую часть этого уравнения, получим

$$\begin{aligned} \alpha_1 &= \tfrac{k_\alpha - (k_\alpha - (k_\alpha \div n) \cdot n)}{n} \bmod n \\ &= \tfrac{(k_\alpha \div n) \cdot n}{n} \bmod n \\ &= (k_\alpha \div n) \bmod n. \end{aligned} \tag{11}$$

По определению операции mod отсюда следует

$$\alpha_1 = k_\alpha \div n - ((k_\alpha \div n) \div n) \cdot n. \tag{12}$$

Подставив в (8) вместо α_0 правую часть уравнения (10), а вместо α_1 – правую часть уравнения (12), получим

$$\begin{aligned} \alpha_2 &= \tfrac{k_\alpha - \alpha_0 - \alpha_1 n}{n^2} \bmod n \\ &= \tfrac{k_\alpha - (k_\alpha - (k_\alpha \div n) \cdot n) - (k_\alpha \div n - ((k_\alpha \div n) \div n) \cdot n) \cdot n}{n^2} \bmod n \\ &= ((k_\alpha \div n) \div n) \bmod n. \end{aligned} \tag{13}$$

Из (11) и (13) для $i = 1, \ldots, n-1$ по индукции получаем:

$$\alpha_i = \underbrace{(((k_\alpha \div n) \ldots) \div n)}_{i} \mathrm{mod} n. \tag{14}$$

Пусть $g = (g_0, \ldots, g_{n-1})$ – нулевая вершина куба следящей области; $y = (y_0, \ldots, y_{n-1})$ – нулевая вершина произвольной ячейки α. Выразим координаты точки y через координаты точки g. Обозначим $s = \frac{r}{K^{1/n}}$ – шаг сетки. Тогда

$$y_i = g_i + s\alpha_i \tag{15}$$

для $i = 0, \ldots, n-1$.

Определим в качестве центральной ячейки куба ячейку γ с целочисленными координатами $(\gamma_0, \ldots, \gamma_{n-1})$, где

$$\gamma_0 = \ldots = \gamma_{n-1} = \left\lfloor K^{1/n} \Big/ 2 \right\rfloor \tag{16}$$

Пусть $q = (q_0, \ldots, q_{n-1})$ – нулевая вершина центральной ячейки γ. Выразим координаты точки q через координаты точки g, используя формулу (15):

$$q_i = g_i + s\gamma_i, \tag{17}$$

для $i = 0, \ldots, n-1$.

Пусть y – нулевая вершина ячейки α.Тогда область внутри ячейки α (включая границы) задается системой из $2n$ неравенств:

$$\begin{cases} -x_0 \le -y_0 \\ \ldots\ldots\ldots\ldots \\ -x_{n-1} \le -y_{n-1} \\ x_0 \le y_0 + s \\ \ldots\ldots\ldots\ldots \\ x_{n-1} \le y_{n-1} + s \end{cases} \tag{18}$$

Чтобы найти пересечение ячейки α с многогранником M необходимо к системе неравенств $Ax \le b$ добавить неравенства из системы (18). При этом получается система из $m + 2n$ неравенств с n неизвестными.

Параллельная версия следящего алгоритма изображена на рис. 2 в виде диаграммы деятельности UML. В качестве начального значения длины ребра r кубической следящей области берется значение R, а в качестве нулевой вершины g кубической следящей области берется точка G, которые заведомо обеспечивают покрытие многогранника M следящей областью. Начальный шаг сетки s определяется по формуле $s = \frac{r}{K^{1/n}}$, где количество ячеек K следящей области вычисляется по формуле (4). Координаты целевой точки z получаются путем умножения вектора c коэффициентов целевой функции на масштабирующий коэффициент T, выбираемый таким образом, чтобы целевая точка находилась вне следящей области. Целочисленные координаты центральной ячейки следящего куба определяется по формуле (16). Нулевая вершина q центральной ячейки следящей области соответственно вычисляется по формуле (17).

Вектор-функция $\pi(z, k)$ с помощью фейеровского процесса, описанного в разделе 2, вычисляет псевдопроекцию целевой точки z на пересечение многогранника с ячейкой, имеющей порядковый номер k, и возвращает точку псевдопроекции x_k или вектор $(-1, \ldots)$,

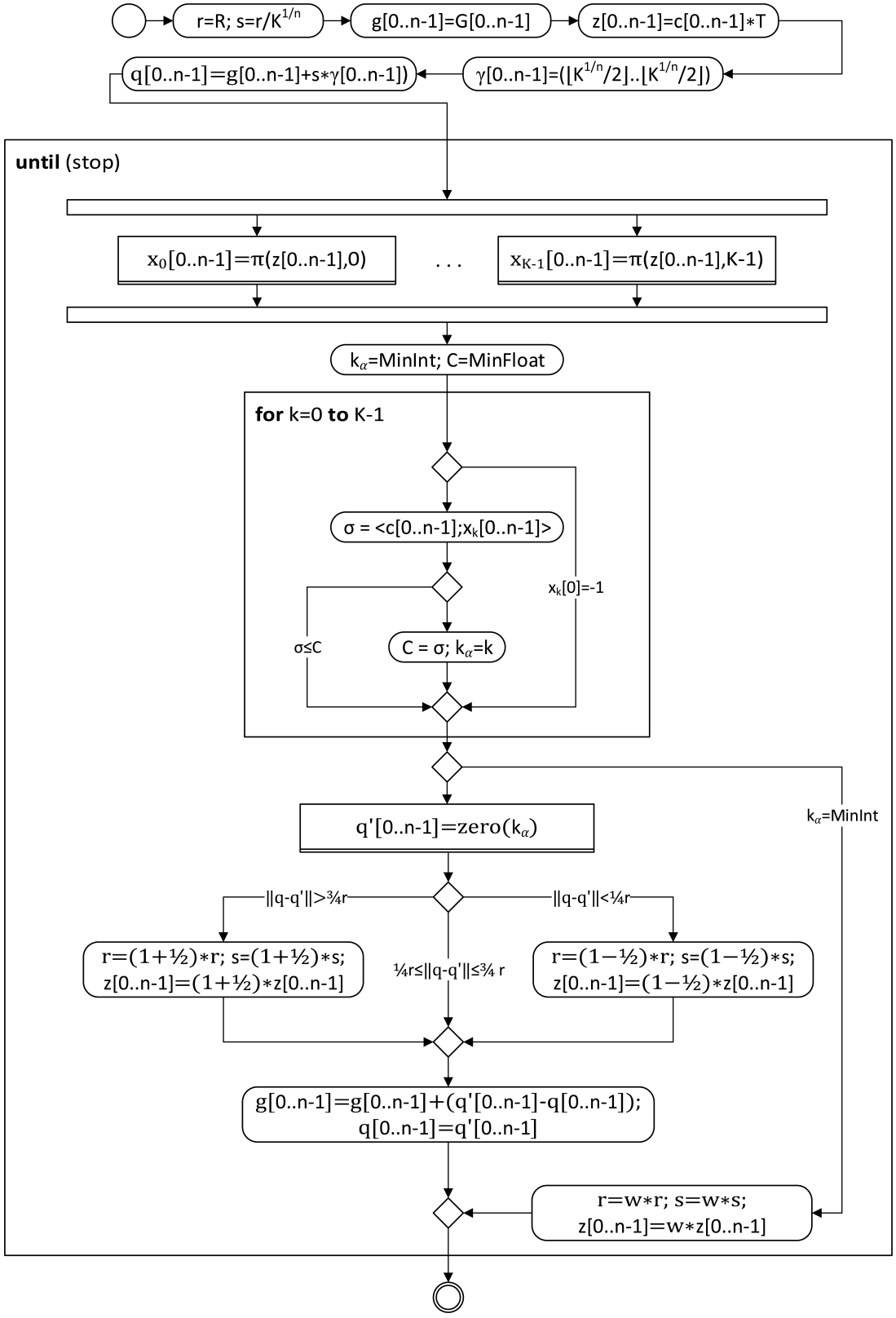

Рис. 2. Параллельная версия следящего алгоритма.

если точка псевдопроекции не принадлежит многограннику (пересечение многогранника M с ячейкой пусто).

Псевдопроекции вычисляются параллельно без обменов данными в цикле **until**. В ходе вычисления псевдопроекций исходные данные задачи (1) могут меняться. После того как все точки псевдопроекций $x_0, \ldots, x_{K-1}$ вычислены, в цикле **for** происходит вычисление порядкового номера k_α ячейки α, на которой достигается максимум целевой функции в точке псевдопроекции. При этом в переменной C вычисляется максимум целевой функции. В соответствии с этим в качестве начального значения k_α выбирается наименьшее целое число в машинном представлении `MinInt`, а в качестве начального значения C выбирается наименьшее вещественное число в машинном представлении `MinFloat`.

После завершения цикла **for** проверяется значение k_α. Если оно равно `MinInt`, значит пересечение всех ячеек с многогранником M оказалось пустым. Это означает, что в результате изменения исходных данных задачи (1) многогранник M «уплыл» за пределы следящей области. В этом случае в w раз увеличиваются: длина r ребра следящей области, шаг сетки s и координаты целевой точки z. Константа w является параметром алгоритма.

Если значение k_α отлично от `MinInt`, значит пересечение многогранника M со следящей областью не пусто. В этом случае в качестве новой центральной ячейки рассматривается ячейка с номером k_α и вычисляются координаты q' – нулевой вершины этой ячейки. Для этого используется вектор-функция `zero`(k_α), диаграмма деятельности которой приведена на рис. 3.

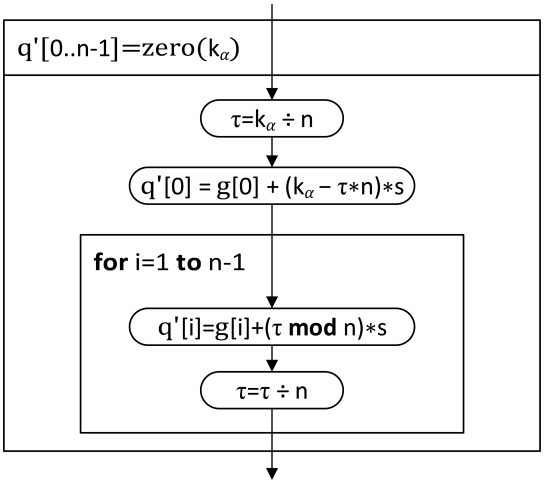

Рис. 3. Вектор-функция, вычисляющая нулевую вершину ячейки с порядковым номером k_α.

Далее рассматриваются три случая в зависимости от удаленности нулевой вершины новой центральной ячейки следящей области от старой. Если $\|q - q'\| > \frac{3}{4}r$, то r, s и координаты z увеличиваются в полтора раза. Если $\|q - q'\| < \frac{1}{4}r$, то r, s и координаты z уменьшаются в полтора раза. Если $\frac{1}{4}r \le \|q - q'\| \le \frac{3}{4}r$ то r, s и координаты z остаются прежними. После этого происходит сдвиг нулевой вершины следящей области на вектор $(q' - q)$ и копирование координат q' в q. Далее выполняется очередная итерация цикла **until**.

5. Заключение

В работе была описана параллельная реализация следящего алгоритм для решения нестационарных задач большой размерности на кластерных вычислительных системах. Алгоритм был реализован на языке Си с использованием библиотеки MPI. Начальные эксперименты на модельной задаче показали почти линейную масштабируемость параллельной реализации следящего алгоритма на 480 узлах суперкомпьютера «Торнадо ЮУрГУ». Авторы

планируют реализовать внутриузловое распараллеливание следящего алгоритма с использованием технологии параллельного программирования OpenMP. Объектом распараллеливания на этом уровне является отдельный фейеровский процесс, вычисляющий псевдопроекцию. Для каждого подвектора организуется отдельная нить. Вычисления предполагается выполнять с использованием ускорителей Xeon Phi, которыми оснащен вычислительный кластер «Торнадо ЮУрГУ». Эффективное использование многоядерных ускорителей достигается путем распараллеливания процесса вычисления отдельной псевдопроекции. При этом предполагается использовать метод разбиения вектора на подвекторы [9]. Суть метода заключается в том, что вектор исходной точки делится на подвекторы по числу процессорных ядер многоядерного ускорителя. На каждом подвекторе независимо делается v фейеровских приближений с использованием редуцированных фейеровских отображений. Такое редуцированное отображение воздействует только на «свой» подвектор, оставляя оставшуюся часть вектора без изменений. Затем нити управления через общую память обмениваются модифицированными подвекторами и процесс продолжается. В работе [9] была доказана сходимость описанного итерационного процесса.

Литература

1. *Еремин И.И., Мазуров Вл.Д.* Нестационарные процессы математического программирования. М.: Наука, 1979. 291 с.
2. *Мазуров Вл.Д.* Распознавание образов как метод формирования плохо формализуемых ограничений в моделях планирования // Оптимизация. Вып. 10(27). Новосибирск: СО АН СССР, 1973. С. 144-158.
3. *Дышаев М.М., Соколинская И.М.* Представление торговых сигналов на основе адаптивной скользящей средней Кауфмана в виде системы линейных неравенств // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. 2013. Т. 2. № 4. С. 103-108.
4. *Ананченко И.В., Мусаев А.А.* Торговые роботы и управление в хаотических средах: обзор и критический анализ // Труды СПИИРАН. 2014. № 3 (34). С. 178-203.
5. *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.
6. *Соколинская И.М., Соколинский Л.Б.* О применении фейеровских отображений в задачах линейной оптимизации с быстро меняющимися входными данными // Информационный бюллетень Ассоциации программирования. № 13. ИММ УрО РАН, 2015. С. 56-58.
7. *Соколинская И.М., Соколинский Л.Б.* Алгоритм решения нестационарных задач линейного программирования для кластерных вычислительных систем с многоядерными ускорителями // Параллельные вычислительные технологии (ПаВТ'2015): труды международной научной конференции. Челябинск: Издательский центр ЮУрГУ, 2015. С. 477-481.
8. *Еремин И.И.* Фейеровские методы для задач выпуклой и линейной оптимизации. Челябинск: Изд-во ЮУрГУ, 2009. 200 с.
9. *Ершова А.В., Соколинская И.М.* О сходимости масштабируемого алгоритма построения псевдопроекции на выпуклое замкнутое множество // Вестник Южно-Уральского государственного университета. Серия: Математическое моделирование и программирование. 2012. № 18 (277). С. 5-12.