Исследование фазы *Quest* алгоритма NSLP для решения нестационарных задач линейного программирования на кластерных вычислительных системах*

Л.Б. Соколинский, И.М. Соколинская

Южно-Уральский государственный университет

Статья посвящена исследованию фазы *Quest* алгоритма NSLP для решения нестационарных задач линейного программирования сверхбольшой размерности на кластерных вычислительных системах. Алгоритм NSLP состоит из двух фаз. В первой фазе, называемой *Quest*, происходит поиск решения системы неравенств, задающих систему ограничений задачи линейного программирования, в условиях динамического изменения исходных данных. Во второй фазе, называемой *Targeting*, формируется специальная система точек, имеющая форму *n*-мерного осесимметричного креста, которая передвигается в *n*-мерном пространстве таким образом, чтобы решение задачи линейного программирования постоянно находилось в ε-окрестности центральной точки креста. В статье дается формальное описание алгоритма NSLP и рассматриваются подходы к реализации фазы *Quest* в условиях динамического изменения исходных данных. В качестве метода для нахождения решения нестационарной системы линейных неравенств используется модифицированный метод Чиммино. На основе этого метода на языке C++ строится параллельная реализация фазы *Quest* с использованием библиотек параллельного программирования OpenMP и MPI. Обсуждаются результаты масштабных вычислительных экспериментов.

Ключевые слова: линейное программирование, нестационарная задача ЛП большой размерности, параллельный алгоритм, алгоритм NSLP, фаза *Quest*, модифицированный алгоритм Чиммино для неравенств, кластерная вычислительная система.

1. Введение

Современные технологии сбора, накопления и обработки больших данных [1] привели к появление математических моделей в виде задач линейного программирования (ЛП) сверхбольшой размерности [2]. В качестве примеров можно привести задачи, возникающие в таких областях, как составление расписаний, логистика, реклама, ритейл, электронная торговля [3], квантовая физика [4], управления пассивами и активами [5], алгоритмическая торговля [6–9] и др. В подобных задачах количество переменных и неравенств в системе ограничений, формируемых с использованием больших данных, может составлять десятки миллионов. Во многих случаях, особенно это характерно для экономико-математического моделирования, указанные задачи ЛП имеют нестационарный характер, когда исходные данные (коэффициенты при переменных в системе ограничений, правые части неравенств, коэффициенты целевой функции) меняются в процессе решения задачи, при этом период изменения исходных данных может находиться в пределах сотых долей секунды.

До настоящего времени одним из самых распространенных способов решения задачи ЛП являлся класс алгоритмов, предложенных и разработанных Данцигом на основе симплекс-метода [10]. Симплекс-метод оказался эффективным для решения большого класса задач ЛП. Однако, в определенных случаях симплекс-методу приходится перебирать все вершины симплекса, что соответствует экспоненциальной временной сложности. Кармаркар в работе [11] предложил

* Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 17-07-00352 а, Правительства РФ в соответствии с Постановлением №211 от 16.03.2013 г. (соглашение № 02.A03.21.0011) и Министерства науки и высшего образования РФ (государственное задание 2.7905.2017/8.9).

алгоритм внутренних точек, который демонстрирует полиномиальное время решения задачи ЛП. В статье [12] продемонстрирован подход к решению сверхбольших задач целочисленного линейного программирования с использованием фреймворка Hadoop [13,14]. В качестве задачи взята реальная математическая модель, используемая для управления движениями самолетов, включающая в себя миллионы переменных. Однако, следует отметить, что фреймворк Hadoop, основанный на вычислительной парадигме MapReduce [15,16], используется только для считывания исходных данных с внешних носителей. Сама задача целочисленного линейного программирования решается традиционными методами.

Симплекс-метод и метод внутренних точек на сегодня остаются основными методами решения задачи ЛП, однако эти методы могут оказаться неэффективными в случае сверхбольших задач ЛП с быстро меняющимися исходными данными. Для решения сверхбольших нестационарных задач линейного программирования авторами в работе [17] был предложен масштабируемый алгоритм *NSLP* (*Non-Stationary Linear Programming*), ориентированный на кластерные вычислительные системы. Алгоритм состоит из двух фаз: *Quest* (поиск) и *Targeting* (позиционирование).

На фазе *Quest* происходит поиск решения системы неравенств, задающих систему ограничений задачи линейного программирования в условиях динамического изменения исходных данных. Исследованию фазы *Quest* были посвящены работы [17–20]. В работе [18] было введено понятие псевдопроекции на выпуклое замкнутое множество, обобщающее понятие проекции. Метод псевдопроекции может быть использован на фазе *Quest*, если в качестве выпуклого множества взять многогранник, определяемый ограничениями задачи ЛП. Для вычисления псевдопроекции использовались фейеровские приближения [21], способные «самоисправляться» в ответ на динамическое изменение исходных данных. В статье [19] было показано, что при вычислении псевдопроекции на многогранники большой размерности могут эффективно использоваться многоядерные ускорители. В [17] была доказана теорема о необходимом условии сходимости итерационного процесса вычисления псевдопроекции в случае, когда изменения исходных данных ограничиваются параллельным переносом многогранника. В работе [20] была показана высокая масштабируемость итерационного алгоритма Чиммино для неравенств на многопроцессорных системах кластерного типа. Алгоритм Чиммино может быть использован на фазе *Quest* алгоритма NSLP вместо метода псевдопроекций в стационарном случае.

На фазе *Targeting* формируется специальная система точек, имеющая форму n-мерного осесимметричного креста, которая передвигается в n-мерном пространстве таким образом, чтобы решение задачи линейного программирования постоянно находилось в ε-окрестности центральной точки креста. Фаза *Targeting* была исследована в работе [22] с помощью модели параллельных вычислений BSF [23,24]. Было показано, что, если в ходе вычислений динамически меняются все исходные данные задачи ЛП, верхняя граница масштабируемости программы будет расти как корень квадратный от размерности задачи. Если же в каждом неравенстве системы ограничений меняется не более одного параметра, то верхняя граница масштабируемости программы будет расти пропорционально размерности задачи.

В данной статье дается формальное описание алгоритма NSLP и рассматриваются подходы к реализации фазы *Quest* в условиях динамического изменения исходных данных. Статья имеет следующую структуру. В разделе 2 дается общая постановка нестационарной задачи ЛП и формальное описание алгоритма NSLP. Раздел 3 посвящен вопросам реализации фазы *Quest*. В разделе 4 приводятся сведения о программной реализации фазы *Quest*, а также описываются результаты масштабных вычислительных экспериментов на кластерной вычислительной системе. В разделе 5 суммируются полученные результаты и намечаются направления дальнейших исследований.

2. Нестационарная задача ЛП и алгоритм NSLP

Пусть в пространстве $\mathbb{R}^n$ задана нестационарная задача ЛП

$$\overline{x} = \arg\max\left\{\left\langle c^{(t)}, x\right\rangle \mid A^{(t)}x \le b^{(t)}\right\}^{1}, \tag{1}$$

где матрица $A^{(t)}$ имеет m строк. Мы здесь предполагаем, что ограничение $x \ge 0$ также включено в систему $A^{(t)}x \le b^{(t)}$ в виде неравенств

$$\begin{array}{ccccccccccc} -x_1 & + & 0 & + & \cdots & \cdots & \cdots & + & 0 & \le & 0; \\ 0 & - & x_2 & + & 0 & + & \cdots & + & 0 & \le & 0; \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & + & \cdots & \cdots & \cdots & + & 0 & - & x_n & \le & 0. \end{array} \tag{2}$$

Нестационарность задачи (1) заключается в том, что значения элементов матрицы $A^{(t)}$ и векторов $b^{(t)}$, $c^{(t)}$ зависят от времени $t \in \mathbb{R}_{\ge 0}$. Конечно, это не относится к неравенствам вида (2). Пусть $M^{(t)}$ – многогранник, задаваемый ограничениями нестационарной задачи ЛП (1) в момент времени t. Такой многогранник всегда является выпуклым замкнутым множеством. Определим расстояние от точки $x \in \mathbb{R}^n$ до многогранника $M^{(t)}$ следующим образом:

$$d\left(x, M^{(t)}\right) = \inf_{y \in M^{(t)}} \|x - y\|. \tag{3}$$

Здесь $\|\cdot\|$ обозначает евклидову норму.

Пусть имеется отображение $\Phi : \mathbb{R}^n \times \mathbb{R}^{mn} \times \mathbb{R}^n \to \mathbb{R}^n$, обладающее следующими свойствами:

$$\Phi\left(y, A^{(t)}, b^{(t)}\right) = y \quad \forall y \in M^{(t)}; \tag{4}$$

$$d\left(\Phi\left(x, A^{(t)}, b^{(t)}\right), M^{(t)}\right) < d\left(x, M^{(t)}\right) \quad \forall x \notin M^{(t)}. \tag{5}$$

С неформальной точки зрения условие (4) означает, что отображение Φ оставляет неподвижной любую точку, принадлежащую многограннику $M^{(t)}$. Условие (5) означает, что любую точку x, не принадлежащую многограннику $M^{(t)}$, отображение Φ переводит в точку, находящуюся ближе к многограннику $M^{(t)}$, чем точка x.

Пусть также имеется отображение $\Psi : \mathbb{R}^n \times \mathbb{R}^{mn} \times \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$, обладающее следующими свойствами:

$$\Psi\left(x, A^{(t)}, b^{(t)}, c^{(t)}\right) = x \quad \forall x \notin M^{(t)}; \tag{6}$$

$$\Psi\left(x, A^{(t)}, b^{(t)}, c^{(t)}\right) = x \quad \forall x \in M^{(t)} : \left\langle c^{(t)}, x\right\rangle = \left\langle c^{(t)}, \overline{x}\right\rangle; \tag{7}$$

$$\left\langle c^{(t)}, \Psi\left(x, A^{(t)}, b^{(t)}, c^{(t)}\right)\right\rangle > \left\langle c^{(t)}, x\right\rangle \quad \forall x \in M^{(t)} : \left\langle c^{(t)}, x\right\rangle < \left\langle c^{(t)}, \overline{x}\right\rangle. \tag{8}$$

С неформальной точки зрения условие (6) означает, что отображение Ψ оставляет неподвижной любую точку, не принадлежащую многограннику $M^{(t)}$. Условие (7) означает, что отображение Ψ оставляет неподвижной любую точку, принадлежащую многограннику $M^{(t)}$, в которой достигается максимум целевой функции. Условие (8) означает, что любую точку x, не принадлежащую многограннику $M^{(t)}$, отображение Ψ переводит в точку, значение целевой функции в которой будет больше, чем в точке x.

[1] $\langle c, x\rangle$ обозначает скалярное произведение двух векторов.

Определим функцию $inM : \mathbb{R}^n \to \{0;1\}$ следующим образом:

$$inM\left(x, A^{(t)}, b^{(t)}\right) = \begin{cases} 0, \text{ if } A^{(t)}x \le b^{(t)} \wedge x \ge 0; \\ 1, \text{ if } A^{(t)}x > b^{(t)} \vee x < 0. \end{cases} \quad (9)$$

То есть, функция inM возвращает значение 1, если точка x принадлежит многограннику $M^{(t)}$, и 0 в противном случае.

Работа алгоритма *NSLP* (*Non-Stationary Linear Programming*) в общем виде может быть описана следующей последовательностью действий.

Алгоритм 1. *NSLP для решения нестационарной задачи ЛП.*

1. $input\left(x^{(0)}\right);\ x := x^{(0)}$
2. $input\left(A^{(t)}, b^{(t)}, c^{(t)}\right)$
3. $x := \Phi\left(x, A^{(t)}, b^{(t)}\right)$
4. **if not** $inM\left(x, A^{(t)}, b^{(t)}\right)$ **goto** 2
5. $\tilde{x} := \Psi\left(x, A^{(t)}, b^{(t)}, c^{(t)}\right)$
6. if $\left\|\tilde{x} - x\right\| < \varepsilon$ $output(\tilde{x})$
7. $x := \tilde{x}$
8. $input\left(A^{(t)}, b^{(t)}, c^{(t)}\right)$
9. **if** $inM\left(x, A^{(t)}, b^{(t)}\right)$ **goto** 5
10. goto 2

На шаге 1 задается начальное приближение $x^{(0)}$. В качестве начального приближения может быть взята произвольная точка. Шаги 2-4 соответствуют фазе *Quest*, представляющей собой итерационный процесс: на шаге 2 осуществляется ввод исходных данных нестационарной задачи линейного программирования, соответствующих текущему моменту времени; на шаге 3 с помощью отображения φ вычисляется следующее приближение к многограннику $M^{(t)}$; на шаге 4 происходит выход из фазы *Quest*, в момент, когда найдена точка x, принадлежащая многограннику $M^{(t)}$. Шаги 5-9 соответствуют фазе *Targeting*: на шаге 5 с помощью отображения ψ вычисляется точка $\tilde{x}$, принадлежащая многограннику M_t, в которой целевая функция принимает большее значение, чем в предыдущей точке x; если на шаге 6 расстояние между точками $\tilde{x}$ и x меньше наперед заданного положительного числа ε, то $\tilde{x}$ выводится как приближенное решение задачи (1) в момент времени t; на шаге 7 координаты точки $\tilde{x}$ присваиваются точке x; на шаге 8 происходит обновление исходных данных задачи (1); если после этого точка x осталась внутри многогранника $M^{(t)}$, то происходит переход на шаг 5, начинающий следующую итерацию фазы *Targeting*. В противном случае осуществляется возврат на фазу *Quest*.

В работе [22] был предложен алгоритм вычисления функции Ψ, используемой на фазе *Targeting*. В следующем разделе мы рассмотрим вариант выбора функции Φ, используемой на фазе *Quest*.

3. Реализация фазы *Quest*

Эффективный способ построения функции Φ, используемой в фазе *Quest*, дает *модифицированный метод Чиммино*. Метод Чиммино был первоначально разработан для решения систем линейных уравнений [25,26]. Его обобщение для произвольных выпуклых множеств было выполнено Ауслендером в работе [27]. Цензор и Эльфвинг в [28,29] предложили модификацию ме-

тода Чиммино для решения систем линейных неравенств в Евклидовом пространстве $\mathbb{R}^n$. *Модифицированный метод Чиммино* начинает свою работу с произвольной точки в $\mathbb{R}^n$ в качестве начального приближения, и затем на каждой итерации вычисляет центр масс множества точек, представляющих собой отражения предыдущего приближения относительно ограничивающих гиперплоскостей. При этом отражения, не противоречащие соответствующим неравенствам, в расчет не принимаются.

Дадим формальное описание этого метода в контексте нестационарной задачи ЛП (1). Обозначим *i*-тое неравенство системы $A^{(t)}x \le b^{(t)}$ следующим образом: $a_i^{(t)}(x) \le b_i^{(t)}$ $(i = 1,\ldots,m)$. Везде далее мы предполагаем, что $a_i^{(t)} \ne 0$. Пусть $P_i^{(t)}$ обозначает полупространство, задаваемое неравенством $a_i^{(t)}(x) \le b_i^{(t)}$. Тогда

$$M^{(t)} = \bigcap_{i=1}^{m} P_i^{(t)}.$$

Каждое уравнение $a_i^{(t)}(x) = b_i^{(t)}$ определяет гиперплоскость $H_i^{(t)}$:

$$H_i^{(t)} = \left\{ x \in \mathbb{R}^n \mid \left\langle a_i^{(t)}, x \right\rangle = b_i^{(t)} \right\}. \tag{10}$$

Ортогональная проекция $\pi_{H_i^{(t)}}(x)$ точки x на гиперплоскость $H_i^{(t)}$ вычисляется по следующей формуле:

$$\pi_{H_i^{(t)}}(x) = x + \frac{b_i^{(t)} - \left\langle a_i^{(t)}, x \right\rangle}{\left\| a_i^{(t)} \right\|^2} a_i^{(t)}. \tag{11}$$

Ортогональное отражение $\rho_{H_i^{(t)}}(x)$ точки x относительно гиперплоскости H_i вычисляется следующим образом:

$$\rho_{H_i^{(t)}}(x) = \pi_{H_i^{(t)}}(x) - x = \frac{b_i^{(t)} - \left\langle a_i^{(t)}, x \right\rangle}{\left\| a_i^{(t)} \right\|^2} a_i^{(t)}. \tag{12}$$

Определим *положительную срезку* ортогонального отражения:

$$\rho_{H_i^{(t)}}^{+}(x) = \frac{\min\left\{ b_i^{(t)} - \left\langle a_i^{(t)}, x \right\rangle, 0 \right\}}{\left\| a_i^{(t)} \right\|^2} a_i^{(t)}. \tag{13}$$

Положим

$$\varphi^{(t)}(x) = x + \frac{\lambda}{m} \sum_{i=1}^{m} \rho_{H_i^{(t)}}^{+}(x) = x + \frac{\lambda}{m} \sum_{i=1}^{m} \frac{\min\left\{ b_i^{(t)} - \left\langle a_i^{(t)}, x \right\rangle, 0 \right\}}{\left\| a_i^{(t)} \right\|^2} a_i^{(t)}, \tag{14}$$

где λ обозначает коэффициент релаксации: $0 < \lambda < 2$. Заметим, что формула (14) совпадает с формулой (3) из работы [17]. *Модифицированный алгоритм Чиммино* для неравенств имеет следующий вид [29].

Алгоритм 2. *Модифицированный алгоритм Чиммино* для неравенств.

1. $k := 0$; $x_0 := \mathbf{0}$.
2. $x_{k+1} := x_k + \frac{\lambda}{m} \sum_{i=1}^{m} \rho_{H_i^{(t)}}^{+}(x_k)$.
3. Если $\left\| x_{k+1} - x_k \right\|^2 < \varepsilon$, перейти на шаг 5.
4. $k := k + 1$; перейти на шаг 2.
5. Стоп.

$$\begin{cases} x_0 & & & & & \le & 200 \\ & x_1 & & & & \le & 200 \\ & & \ddots & & & \cdots & \cdots \\ & & & & x_{n-1} & \le & 200 \\ x_0 & + x_1 & \cdots & + & x_{n-1} & \le & 200(n-1)+100 \\ x_0 & + x_1 & \cdots & + & x_{n-1} & \ge & 100 \\ x_0 & & & & & \ge & 0 \\ & x_1 & & & & \ge & 0 \\ & & \ddots & & & \cdots & \cdots \\ & & & & x_{n-1} & \ge & 0 \end{cases}$$

Рис. 1. Масштабируемая система неравенств.

Указанный алгоритм вычисляет точку, принадлежащую многограннику $M^{(t)}$. Если многогранник $M^{(t)}$ не меняется со временем, алгоритм 2 завершает свою работу за конечное число шагов при любом $\varepsilon > 0$. Это вытекает из того, что отображение $\varphi^{(t)}$, используемое на шаге 2, при фиксированном t является однозначным непрерывным $M^{(t)}$-фейеровским[1] отображением для коэффициента релаксации $0 < \lambda < 2$ [21] и, следовательно, итерационный процесс, используемый в алгоритме 2, сходится к точке, принадлежащей многограннику $M^{(t)}$ [30]. Еще одно доказательство сходимости алгоритма 2 можно найти в [29]. Однако оба этих доказательства справедливы только для стационарных задач. Нестационарный случай был рассмотрен нами в работе [17] в предположении, что изменения исходных данных ограничиваются параллельным переносом многогранника $M^{(t)}$. Для этого случая авторами была доказана теорема о достаточном условии сходимости итерационного процесса, порождаемого алгоритмом 2.

Положим

$$\Phi\left(x, A^{(t)}, b^{(t)}\right) = \varphi^{(t)}(x). \tag{15}$$

Тогда выполняются свойства (4) и (5). Следовательно, функция $\varphi^{(t)}$ может быть использована в алгоритме 1.

4. Программная реализация и вычислительные эксперименты

Алгоритм 2 для нестационарной задачи был нами реализован на языке C++ с использованием программного каркаса BSF [24] и библиотек параллельного программирования OpenMP и MPI. Схема параллельной реализации идентична схеме реализации алгоритма Чиммино, детально описанной в статье [20]. Исходные коды свободно доступны в сети Интернет по адресу https://github.com/leonid-sokolinsky/NSLP-*Quest*. В качестве задачи была использована масштабируемая система неравенств размерности n из статьи [19] (см. рис. 1). Количество неравенств m в этой системе вычисляется по формуле $m = 2n + 2$. Нестационарность моделировалась путем смещения многогранника $M^{(t)}$ по одной из координатных осей с различной скоростью. Вычислительные эксперименты проводились на вычислительном кластере «Торнадо ЮУрГУ» [31]. Результаты экспериментов представлены на рис. 2 и рис. 3. На диаграммах показана зависимость времени работы алгоритма 2 от скорости смещения многогранника $M^{(t)}$. В первом эксперименте

[1] Однозначное отображение $\varphi : \mathbb{R}^n \to \mathbb{R}^n$ называется фейеровским относительно множества M или кратко *M*-фейеровским, если

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

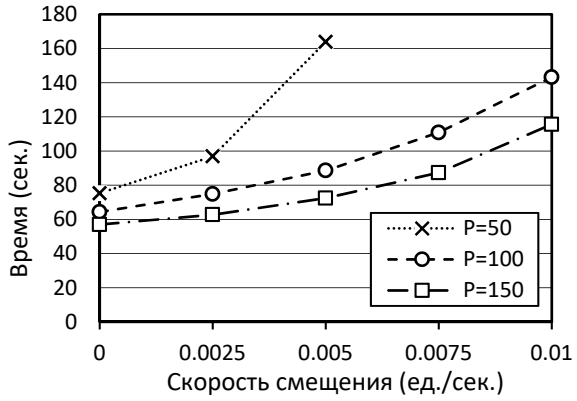

Рис. 2. Время работы алгоритма 2 при n = 32000, m = 64002 (P – количество процессорных узлов).

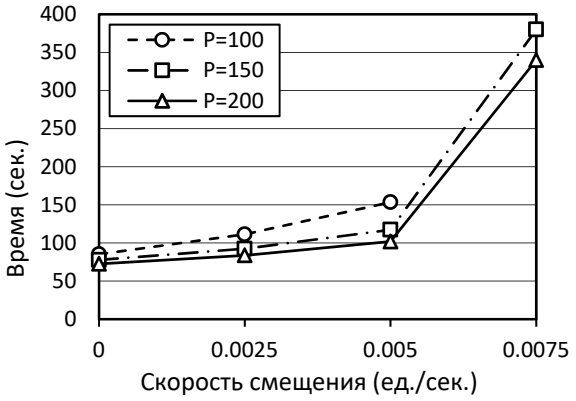

Рис. 3. Время работы алгоритма 2 при n = 54000, m = 108002 (P – количество процессорных узлов).

(рис. 2) использовалась система размерности $n = 34000$ с количеством неравенств $m = 64002$. Скорость смещения многогранника менялась от 0 до 0.01 единицы в секунду с шагом 0.0025. Количество вычислительных узлов P, на которых запускалась задача, варьировалось от 50 до 150. Эксперимент показал, что при $P = 50$ вычислительный процесс начинает отставать от многогранника при скорости, большей 0.005 ед./сек. При $P = 100$ предельный порог скорости смещения увеличивается до 0.01 ед./сек. Для $P = 100$ получаем тот же порог, но при меньшем времени вычислений. Дальнейшее увеличение количества узлов не преодолевало указанный порог и не приводило к заметному росту производительности. Во втором эксперименте (рис. 3) использовалась система размерности $n = 54000$ с количеством неравенств $m = 108002$. Скорость смещения многогранника также менялась от 0 до 0.01 единицы в секунду с шагом 0.0025. Эксперимент показал, что при $P = 50$ вычислительный процесс начинает отставать от многогранника уже при скорости, равной 0.0025 ед./сек. При $P = 100$ предельный порог скорости равен 0.005 ед./сек. Для $P = 150$ порог скорости составил 0.0075 ед./сек. Аналогичный результат показала конфигурация с количеством процессорных узлов $P = 200$, но при меньшем времени вычислений. Дальнейшее увеличение количества узлов не преодолевало порог в 0.0075 ед./сек. и не приводило к заметному росту производительности. Таким образом, можно сделать вывод, что параллельная реализация алгоритма 2 масштабируется с ростом задачи, однако данный алгоритм не допускает очень быстрых изменений исходной системы неравенств. В соответствии с этим остается актуальной задача разработки более быстрых итерационных алгоритмов для решения нестационарных систем неравенств большой размерности.

5. Заключение

В статье дано формальное описание алгоритма NSLP для решения нестационарных задач линейного программирования большой размерности. Алгоритм NSLP состоит из двух фаз. В первой фазе, называемой *Quest*, происходит поиск решения системы неравенств, задающих систему ограничений задачи линейного программирования, в условиях динамического изменения исходных данных. Во второй фазе, называемой *Targeting*, формируется специальная система точек, имеющая форму n-мерного осесимметричного креста, которая передвигается в n-мерном пространстве таким образом, чтобы решение задачи линейного программирования постоянно находилось в ε-окрестности центральной точки креста. Для фазы *Quest* предложена реализация на основе модифицированного метода Чиммино для неравенств. Указанный метод реализован в виде программы на языке C++ с использованием библиотек параллельного программирования OpenMP и MPI. Данная программа моделирует нестационарность системы линейных неравенств путем смещения многогранника, задающего допустимое множество точек, по одной из координатных осей. Описаны вычислительные эксперименты по решению больших нестационарных систем неравенств на кластерной вычислительной системе. Проведенные эксперименты показали, что параллельный алгоритм на основе модифицированного метода Чиммино хорошо мас-

штабируется с ростом размерности нестационарной задачи, однако данный алгоритм не допускает быстрых динамических изменений в исходной системе неравенств. В качестве направлений дальнейших исследований предполагаются следующие: разработка и исследование эффективной параллельной реализации фазы *Targeting*; поиск и исследование более быстрых методов для реализации фазы *Quest*.

Литература

1. Jagadish H. V. et al. Big data and its technical challenges // Communications of the ACM. ACM, 2014. Vol. 57, no. 7. P. 86–94. DOI:10.1145/2611567.
2. Chung W. Applying large-scale linear programming in business analytics // 2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM). IEEE, 2015. P. 1860–1864. DOI:10.1109/IEEM.2015.7385970.
3. Tipi H. Solving Super-Size Problems with Optimization [Electronic resource]. 2010. URL: http://nymetro.chapter.informs.org/prac_cor_pubs/06-10 Horia Tipi SolvingLargeScaleXpress.pdf (accessed: 21.04.2017).
4. Gondzio J. et al. Solving large-scale optimization problems related to Bell's Theorem // Journal of Computational and Applied Mathematics. Elsevier B.V., 2014. Vol. 263. P. 392–404. DOI:10.1016/j.cam.2013.12.003.
5. Sodhi M.S. LP modeling for asset-liability management: A survey of choices and simplifications // Operations Research. 2005. Vol. 53, no. 2. P. 181–196. DOI:10.1287/opre.1040.0185.
6. Brogaard J., Hendershott T., Riordan R. High-Frequency Trading and Price Discovery // Review of Financial Studies. 2014. Vol. 27, no. 8. P. 2267–2306. DOI:10.1093/rfs/hhu032.
7. Budish E., Cramton P., Shim J. The High-Frequency Trading Arms Race: Frequent Batch Auctions as a Market Design Response // The Quarterly Journal of Economics. 2015. Vol. 130, no. 4. P. 1547–1621. DOI:10.1093/qje/qjv027.
8. Gomber P. et al. High-Frequency Trading // SSRN Electronic Journal. 2011. P. 86. DOI:10.2139/ssrn.1858626.
9. Hendershott T., Jones C.M., Menkveld A.J. Does Algorithmic Trading Improve Liquidity? // The Journal of Finance. 2011. Vol. 66, no. 1. P. 1–33. DOI:10.1111/j.1540-6261.2010.01624.x.
10. Dantzig G.B. Linear programming and extensions. Princeton, N.J.: Princeton university press, 1998. 656 p.
11. Karmarkar N. A new polynomial-time algorithm for linear programming // Combinatorica. 1984. Vol. 4, no. 4. P. 373–395. DOI:10.1007/BF02579150.
12. Cao Y., Sun D. Large-Scale and Big Optimization Based on Hadoop // Studies in Big Data. Vol. 18. Big Data Optimization: Recent Developments and Challenges / ed. Emrouznejad A. Springer, Cham, 2016. P. 375–389. DOI:10.1007/978-3-319-30265-2_16.
13. Mone G., Gregory. Beyond Hadoop // Communications of the ACM. ACM, 2013. Vol. 56, no. 1. P. 22–24. DOI:10.1145/2398356.2398364.
14. Gunther N.J., Puglia P., Tomasette K. Hadoop superlinear scalability // Communications of the ACM. ACM, 2015. Vol. 58, no. 4. P. 46–55. DOI:10.1145/2719919.
15. Dean J., Ghemawat S. MapReduce: Simplified Data Processing on Large Clusters // Proceedings of 6th Symposium on Operating Systems Design & Implementation (December 6-8, 2004, San Francisco, California, USA). Berkeley, CA, USA: USENIX Association, 2004. P. 137–150.
16. Li R. et al. MapReduce Parallel Programming Model: A State-of-the-Art Survey // International Journal of Parallel Programming. Springer US, 2016. Vol. 44, no. 4. P. 832–866. DOI:10.1007/s10766-015-0395-0.
17. Соколинская И.М., Соколинский Л.Б. О решении задачи линейного программирования в эпоху больших данных // Параллельные вычислительные технологии – XI международная конференция, ПаВТ'2017, г. Казань, 3–7 апреля 2017 г. Короткие статьи и описания плакатов. Челябинск: Издательский центр ЮУрГУ, 2017. С. 471–484.

18. Ершова А.В., Соколинская И.М. О сходимости масштабируемого алгоритма построения псевдопроекции на выпуклое замкнутое множество // Вестник ЮУрГУ. Серия: Математическое моделирование и программирование. 2011. № 37(254). С. 12–21.
19. Соколинская И.М., Соколинский Л.Б. Модифицированный следящий алгоритм для решения нестационарных задач линейного программирования на кластерных вычислительных системах с многоядерными ускорителями // Суперкомпьютерные дни в России: труды международной конференции (26-27 сентября 2016 г., г. Москва). Москва: Изд-во МГУ, 2016. С. 294–306.
20. Соколинская И.М., Соколинский Л.Б. Исследование масштабируемости модифицированного алгоритма Чиммино для линейных неравенств // Суперкомпьютерные дни в России: Труды международной конференции (24-25 сентября 2018 г., г. Москва). Москва: Изд-во МГУ, 2018. С. 673–983.
21. Ерёмин И.И. Фейеровские методы для задач выпуклой и линейной оптимизации. Челябинск: Издательский центр ЮУрГУ, 2009. 199 с.
22. Соколинская И.М., Соколинский Л.Б. Масштабируемый алгоритм для решения нестационарных задач линейного программирования // Вычислительные методы и программирование: новые вычислительные технологии. 2018. Т. 19, № 4. С. 540–550. DOI:10.26089/NumMet.v19r448.
23. Ежова Н.А., Соколинский Л.Б. BSF: модель параллельных вычислений для многопроцессорных систем с распределенной памятью // Параллельные вычислительные технологии – XII международная конференция, ПаВТ’2018, г. Ростов-на-Дону, 2–6 апреля 2018 г. Короткие статьи и описания плакатов. Челябинск: Издательский центр ЮУрГУ, 2018. С. 253–265.
24. Ежова Н.А., Соколинский Л.Б. Модель параллельных вычислений для многопроцессорных систем с распределенной памятью // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2018. Т. 7, № 2. С. 32–49. DOI:10.14529/cmse180203.
25. Cimmino G. Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari // La Ricerca Scientifica, XVI, Series II, Anno IX, 1. 1938. P. 326–333.
26. Galántai A. Projectors and Projection Methods. Advances in Mathematics, vol. 6. Boston, MA: Springer US, 2004. 287 p. DOI:10.1007/978-1-4419-9180-5.
27. Auslender A. Optimisation: Methodes Numeriques. Paris, France: Masson, 1976.
28. Censor Y., Elfving T. New methods for linear inequalities // Linear Algebra and its Applications. North-Holland, 1982. Vol. 42. P. 199–211. DOI:10.1016/0024-3795(82)90149-5.
29. Censor Y. et al. New Methods for Linear Inequalities // SIAM Journal on Scientific Computing. Society for Industrial and Applied Mathematics, 2008. Vol. 30, no. 1. P. 473–504. DOI:10.1137/050639399.
30. Ерёмин И.И. Методы фейеровских приближений в выпуклом программировании // Математические заметки. 1968. Vol. 3, no. 2. С. 217–234.
31. Kostenetskiy P.S., Safonov A.Y. SUSU Supercomputer Resources // Proceedings of the 10th Annual International Scientific Conference on Parallel Computing Technologies (PCT 2016). CEUR Workshop Proceedings. Vol. 1576. 2016. P. 561–573.
32. Sokolinsky L.B. Analytical Estimation of the Scalability of Iterative Numerical Algorithms on Distributed Memory Multiprocessors // Lobachevskii Journal of Mathematics. 2018. Vol. 39, no. 4. P. 571–575. DOI:10.1134/S1995080218040121.