

УДК 519.6

## ПАРАЛЛЕЛЬНЫЙ АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ СИЛЬНОЙ ОТДЕЛИМОСТИ НА ОСНОВЕ ФЕЙЕРОВСКИХ ОТОБРАЖЕНИЙ

А. В. Ершова<sup>1</sup>, И. М. Соколинская<sup>1</sup>

Рассматривается задача разделения двух выпуклых непересекающихся многогранников слоем наибольшей толщины. Предлагается параллельный алгоритм решения задачи сильной отделимости на основе фейеровских отображений, допускающий эффективную реализацию на многопроцессорных системах с массовым параллелизмом. Приводятся результаты вычислительных экспериментов, подтверждающие эффективность предложенного подхода. Работа выполнена при финансовой поддержке РФФИ (код проекта 09-01-00546а).

**Ключевые слова:** сильная отделимость, фейеровские отображения, параллельное программирование, псевдопроекция, итерационный процесс, распознавание образов.

**1. Введение.** Задача сильной отделимости состоит в нахождении слоя наибольшей толщины, разделяющего два выпуклых непересекающихся многогранника. Задача сильной отделимости имеет большое значение теоретического и прикладного характера в распознавании образов и включает в себя задачи дискриминации, классификации и др. Хорошо известен итерационный метод решения задачи сильной отделимости, использующий операцию проектирования. Однако на практике применение этого метода существенно ограничивается тем, что далеко не всегда удается построить конструктивную формулу для вычисления проекции точки на выпуклое множество, поэтому целесообразно заменить операцию проектирования последовательностью фейеровских отображений [4]. Указанный метод был описан в работе [3].

Среди современных методов классификации и распознавания образов одно из ведущих мест занимает метод опорных векторов, известный также как метод обобщенного портрета или машины опорных векторов (Support Vector Machines, SVM) [8]. Системы, разработанные на его основе, успешно решают задачи в таких областях, как биоинформатика, машинное зрение, категоризация текстов, распознавание рукописных символов и др. Однако метод опорных векторов оказывается неэффективным в случае изменяющихся исходных данных. Алгоритмы разделения многогранников на основе фейеровских отображений обладают тем преимуществом по сравнению с этим и другими известными методами, что они применимы к нестационарным задачам, т.е. к задачам, в которых исходные данные могут меняться в процессе решения задачи. Такими нестационарными задачами являются, например, задача о портфеле ценных бумаг и задача о спам-фильтре.

При решении задачи о портфеле ценных бумаг [12] является важным вопрос о выборе ценной бумаги из их многообразия при формировании портфеля. Проблему выбора можно решить, условно разделив все ценные бумаги на две части: прибыльные и убыточные. Бумага, которая будет дорожать, считается прибыльной, а бумага, которая будет дешеветь, — убыточной. Каждая ценная бумага представляется в виде многомерной точки, координаты которой — это параметрическое описание конкретной бумаги [1]. Эксперт брокерской компании определяет набор параметров, исходя из которых он может предсказать, будет дорожать или дешеветь бумага. Такими параметрами могут быть положение бумаги в тренде, средний оборот торгов, спрэд между ценами спроса и предложения и др. Количество параметров определяет размерность пространства решаемой задачи. Эксперт формирует два множества точек: множество прибыльных бумаг и множество убыточных бумаг. Для каждого множества строится выпуклая оболочка в виде многогранника. Так мы приходим к задаче сильной отделимости. Построив слой наибольшей толщины, разделяющий два многогранника, мы получаем инструмент, позволяющий автоматически разделять ценные бумаги на прибыльные и убыточные.

Этот же подход может быть использован для создания программы-робота, которая будет в автоматическом режиме совершать биржевые операции, обеспечивая стабильную доходность. В настоящее время на мировых биржах уже трудятся сотни программ-роботов. Это явление получило название “*роботрейдинг*” (*алгоритмическая торговля*) [7]. Эффективный алгоритм разделения многогранников на основе

<sup>1</sup> Южно-Уральский государственный университет, факультет вычислительной математики и информатики, пр. Ленина, 76, 454080, г. Челябинск; А. В. Ершова, преподаватель, e-mail: ershovaav@gmail.com; И. М. Соколинская, доцент, e-mail: irinasokolinsky@gmail.com

фейеровских отображений, адаптирующийся к динамическим изменениям биржевой ситуации, позволит создать программу-робота нового поколения, которая сможет быстро реагировать на изменяющиеся условия рынка ценных бумаг и валют, обеспечивая владельцу преимущество в конкурентной борьбе.

В задаче о спам-фильтре [13] мы должны для каждого электронного письма определить, является ли данное письмо “спамом” или нет. Для этого также вводится система метрик. Пользователь разделяет свои письма и отмечает те, которые считает спамом [11]. На основании этой информации строятся выпуклые оболочки в виде систем линейных неравенств. Первая система задает множество точек-писем, определяемых как спам, вторая система — множество точек-писем, определяемых как неспам. Построив слой наибольшей толщины, разделяющий два многогранника, мы получаем инструмент, позволяющий автоматически разделять письма на “хорошие” и “плохие”. Когда приходит новое письмо, считываются характеристики этого письма, получается точка в рассматриваемом пространстве. Если данная точка попадает в “плохое” полупространство, мы делаем предположение, что это спам; если же в “хорошее”, то предполагаем, что это неспам. Если точка попадает внутрь слоя, то письмо доставляется пользователю с пометкой “возможно, спам”.

При решении практических задач экономико-математического моделирования часто встречаются задачи с большим количеством переменных и ограничений. Размер типичной средней задачи может составлять 20 000 переменных и 5000 ограничений [10]. В отдельных случаях количество переменных может превышать 100 000, а количество ограничений — 20 000. Подобные задачи при решении требуют значительных вычислительных мощностей. В связи с этим актуальной является задача разработки параллельного алгоритма разделения выпуклых многогранников с помощью фейеровских отображений, допускающего эффективную реализацию на многопроцессорных системах с массовым параллелизмом.

В настоящей статье предлагается и исследуется новый масштабируемый параллельный алгоритм для решения задачи сильной отделимости, основанный на использовании фейеровских отображений. Статья организована следующим образом. В разделе 2 строится математическая модель задачи сильной отделимости; приводится итерационный алгоритм  $\mathfrak{F}$  решения задачи сильной отделимости, использующий операции проектирования точки на многогранник; конструируется альтернативный алгоритм  $\mathfrak{F}$ , строящий псевдопроекции путем фейеровских приближений. В разделе 3 предлагается новый масштабируемый алгоритм  $\mathfrak{S}$  построения псевдопроекции на многогранник, в основе которого лежит метод разбиения пространства на подпространства. В разделе 4 описывается параллельная реализация предложенного алгоритма решения задачи сильной отделимости, допускающая эффективное масштабирование на тысячи процессоров. В разделе 5 приводятся результаты численных экспериментов, выполненных на высокопроизводительном вычислительном кластере, позволяющие определить оптимальные значения параметров алгоритма и подтверждающие эффективность предложенных подходов. В заключение суммируются полученные результаты и приводятся направления дальнейших исследований.

**2. Математическая модель.** Пусть даны два выпуклых непересекающихся многогранника  $M \subset \mathbb{R}^n$  и  $N \subset \mathbb{R}^n$ , заданные системами линейных неравенств

$$M = \{x | Ax \leq b\} \neq \emptyset; \quad N = \{x | Bx \leq d\} \neq \emptyset; \quad M \cap N = \emptyset. \quad (1)$$

*Задача сильной отделимости* — это задача нахождения слоя наибольшей толщины ( $\pi$ -слоя), разделяющего  $M$  и  $N$ . Сильная отделимость, по существу, эквивалентна задаче отыскания расстояния между  $M$  и  $N$  в смысле метрики

$$\rho(M, N) = \min \{ \|x - y\| \mid x \in M, y \in N \}. \quad (2)$$

Если  $\bar{x} \in M$  и  $\bar{y} \in N$  являются  $\arg$ -точками задачи (2), т.е.  $\rho(M, N) = \|\bar{x} - \bar{y}\|$ , то слоем наибольшей толщины, разделяющим множества  $M$  и  $N$ , является  $P := \{x \mid x \in P_1 \cap P_2\}$ , где  $P_1$  и  $P_2$  — полупространства, задаваемые линейными неравенствами  $\langle x - \bar{x}, \bar{x} - \bar{y} \rangle \leq 0$  и  $\langle y - \bar{y}, \bar{x} - \bar{y} \rangle \geq 0$ . Таким образом, в краткой форме задачу сильной отделимости можно записать в виде

$$\{\bar{x}, \bar{y}\} \in \text{Arg min} \{ \|x - y\| \mid x \in M, y \in N \}. \quad (3)$$

Задача сильной отделимости может быть решена с помощью известного метода *последовательного проектирования* [2], на основе которого мы можем построить следующий итерационный алгоритм, использующий операции проектирования.

**Алгоритм  $\mathfrak{F}$ .** Пусть даны два выпуклых непересекающихся многогранника  $M \subset \mathbb{R}^n$  и  $N \subset \mathbb{R}^n$ , заданные системами линейных неравенств (1). Обозначим через  $\pi_M$  и  $\pi_N$  операции проектирования точки на многогранники  $M$  и  $N$  соответственно. Пусть задано произвольное начальное приближение  $w_0 \in \mathbb{R}^n$ . Зафиксируем положительное вещественное число  $\varepsilon$ . Алгоритм состоит из следующих шагов:

Шаг 0.  $k := 0$ .

Шаг 1.  $x_{k+1} := \pi_M(w_k)$ .

Шаг 2.  $y_{k+1} := \pi_N(w_k)$ .

Шаг 3.  $w_{k+1} := \frac{1}{2}(x_{k+1} + y_{k+1})$ .

Шаг 4.  $k := k + 1$ .

Шаг 5. Если  $\min \{ \|x_{k+1} - x_k\|, \|y_{k+1} - y_k\| \} \geq \varepsilon$ , то перейти на шаг 1.

Шаг 6. Стоп.

Известно [2], что для последовательностей  $\{x_k\}$  и  $\{y_k\}$ , порождаемых алгоритмом, имеет место сходимость к циклу неподвижности:  $\{x_k\}_{k=1}^{+\infty} \rightarrow \bar{x}$  и  $\{y_k\}_{k=1}^{+\infty} \rightarrow \bar{y}$ , где  $\bar{x} \in M$  и  $\bar{y} \in N$  являются *arg-точками* задачи (2). Следовательно, алгоритм  $\mathfrak{P}$  при  $\varepsilon \rightarrow 0$  позволяет получить приближенное численное решение задачи сильной отделимости. Работа алгоритма  $\mathfrak{P}$  для размерности  $n = 2$  схематично изображена на рис. 1. Если множества  $M$  и  $N$  достаточно просты в смысле простоты реализации операции проектирования точек на них, то алгоритм  $\mathfrak{P}$  может быть использован на практике. Однако если  $M$  и  $N$  — произвольные многогранники, то алгоритм  $\mathfrak{P}$  не может быть признан эффективным, так как не известен универсальный конструктивный метод построения проекции точки на многогранник. Ситуацию можно исправить, если вместо операции проектирования использовать *фейеровские отображения*.

Дадим определение фейеровского отображения.

Пусть  $\varphi \in \{\mathbb{R}^n \rightarrow \mathbb{R}^n\}$ . Отображение  $\varphi$  называется *M-фейеровским*, если выполняются следующие два условия:  $\varphi(y) = y \ \forall y \in M$  и  $\|\varphi(x) - y\| < \|x - y\| \ \forall y \in M, \forall x \notin M$ . Класс *M-фейеровских* отображений будем обозначать через  $\mathbf{F}_M$ . Пусть задано однозначное отображение  $\varphi \in \mathbf{F}_M$ . Под степенью  $\varphi^s(x)$  отображения  $\varphi(x)$  будем понимать его последовательное применение  $s$  раз, т.е.  $\varphi^s(x) = \underbrace{\varphi \dots \varphi}_s(x)$ .

Под *фейеровским процессом*, порождаемым однозначным фейеровским отображением  $\varphi$  при произвольном начальном приближении  $x_0 \in \mathbb{R}^n$ , будем понимать последовательность  $\{\varphi^s(x_0)\}_{s=0}^{+\infty}$ . Известно [2] (лемма 39.1), что в случае, когда однозначное *M-фейеровское* отображение  $\varphi$  является непрерывным, фейеровский процесс сходится к точке, принадлежащей множеству  $M$ :  $\{\varphi^s(x_0)\}_{s=0}^{+\infty} \rightarrow \bar{x} \in M$ .

Для сходящегося фейеровского процесса при произвольном начальном приближении  $x_0$  введем следующее обозначение:  $\lim_{s \rightarrow \infty} \varphi^s(x_0) = \bar{x}$ . Это означает, что для любого вещественного  $\varepsilon > 0$  существует целое неотрицательное число  $S$ , такое, что для всех  $s > S$  имеем  $\|x_s - \bar{x}\| < \varepsilon$ .

**Определение.** Пусть задано однозначное непрерывное отображение  $\varphi \in \mathbf{F}_M$ . Под  $\varphi$ -проектированием (*псевдопроектированием*) точки  $x \in \mathbb{R}^n$  на множество  $M$  мы будем понимать отображение  $\pi_M^\varphi : \mathbb{R}^n \rightarrow M$ , задаваемое соотношением  $\pi_M^\varphi(x) = \lim_{s \rightarrow \infty} \varphi^s(x)$ . Точку  $\pi_M^\varphi(x)$  будем называть *псевдопроекцией* точки  $x$  на множество  $M$ .

Пусть в контексте задачи сильной отделимости (3) заданы два однозначных непрерывных фейеровских отображения  $\varphi \in \mathbf{F}_M$  и  $\psi \in \mathbf{F}_N$ . Используя операции  $\varphi$ - и  $\psi$ -проектирования, мы можем построить следующий алгоритм  $\mathfrak{F}$ , решающий задачу сильной отделимости с использованием фейеровских отображений.

**Алгоритм  $\mathfrak{F}$ .** Пусть задано произвольное начальное приближение  $w_0 \in \mathbb{R}^n$ . Зафиксируем положительное вещественное число  $\varepsilon$ . Алгоритм состоит из следующих шагов.

Шаг 0.  $k := 0$ .

Шаг 1.  $x_{k+1} := \pi_M^\varphi(w_k)$ .

Шаг 2.  $y_{k+1} := \pi_N^\psi(w_k)$ .

Шаг 3.  $w_{k+1} := (x_{k+1} + y_{k+1})/2$ .

Шаг 4.  $k := k + 1$ .

Шаг 5. Если  $\|w_{k+1} - w_k\| \geq \varepsilon$ , то перейти на шаг 1.

Шаг 6. Стоп.

Алгоритм  $\mathfrak{F}$  исследован в работе [5] для разных видов фейеровских отображений. Проведенные вычислительные эксперименты на модельных и случайных задачах подтвердили его эффективность. Однако для больших размерностей работа алгоритма  $\mathfrak{F}$  требовала значительного времени. Например, при размер-

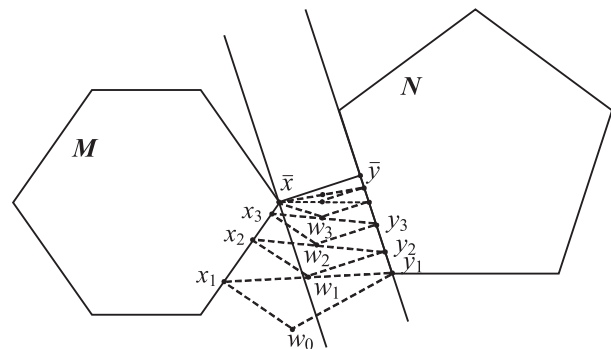


Рис. 1. Работа алгоритма  $\mathfrak{P}$

ности задачи  $n = 512$  время счета на одном процессорном ядре составило 16 часов, а при размерности задачи  $n = 1024$  составило 608 часов (более 25 дней). В связи с этим возникла необходимость разработки параллельной версии этого алгоритма для многопроцессорных систем с массовым параллелизмом. Очевидно, что в алгоритме  $\mathfrak{F}$  ресурсоемкими являются шаги 1 и 2. На каждом из этих шагов реализуется *последовательный* фейеровский процесс, в результате которого мы получаем *псевдопроекцию* точки на многогранник. Проведенные нами вычислительные эксперименты показали, что подобные фейеровские процессы не допускают эффективного распараллеливания на большом количестве процессорных узлов (предел масштабируемости в экспериментах не превышал 8–16 узлов). В следующем разделе мы предложим новый *масштабируемый* алгоритм построения псевдопроекции точки на многогранник с использованием фейеровских отображений, который допускает эффективное распараллеливание на большом количестве процессоров.

**3. Масштабируемый алгоритм  $\mathfrak{S}$  построения псевдопроекции.** В основе масштабируемого алгоритма построения псевдопроекции на многогранник лежит метод разбиения пространства на подпространства. Для каждого подпространства организуется независимый фейеровский процесс. Через каждые  $s$  шагов результаты, полученные на подпространствах, соединяются в один вектор, который и является очередным приближением. Если расстояние между соседними приближениями меньше заданного положительного числа  $\varepsilon$ , то полученный вектор принимается в качестве псевдопроекции. В противном случае вычисления продолжаются.

Дадим формальное описание алгоритма построения псевдопроекции на выпуклый многогранник, допускающего эффективное распараллеливание на большом количестве процессорных узлов. Введем следующие обозначения. Для произвольного линейного подпространства  $\mathbb{P} \subset \mathbb{R}^n$  через  $\pi_{\mathbb{P}}(x)$  будем обозначать ортогональную проекцию  $x \in \mathbb{R}^n$  на линейное подпространство  $\mathbb{P}$ . Везде далее линейное подпространство мы будем называть просто подпространством. Через  $\rho(\mathbb{P}, x) := \min_{p \in \mathbb{P}} \|p - x\|$  будем обозначать расстояние от точки  $x$  до подпространства  $\mathbb{P}$ . Пусть линейное многообразие  $\mathbb{L}$  получается из  $\mathbb{P}$  сдвигом на некоторый вектор  $z$ :  $\mathbb{L} = \mathbb{P} + z$ . Через  $\pi_{\mathbb{L}}(x)$  будем обозначать ортогональную проекцию  $x \in \mathbb{R}^n$  на линейное многообразие  $\mathbb{L}$ :  $\pi_{\mathbb{L}}(x) = \pi_{\mathbb{P}}(x) + z$ . Теперь мы готовы к формальному описанию алгоритма.

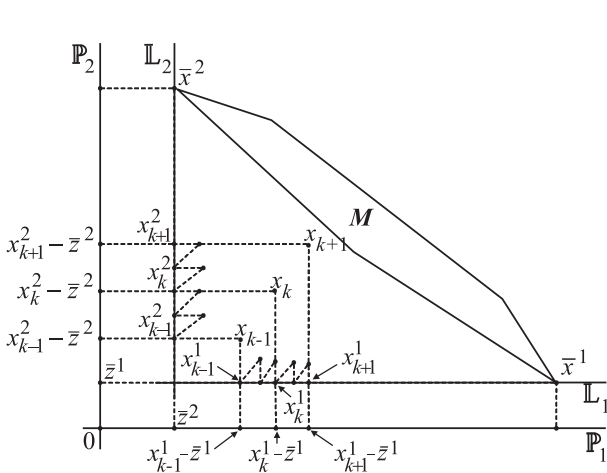


Рис. 2. Работа алгоритма  $\mathfrak{S}$ :  $x_k^1 = \pi_{\mathbb{L}_1}(x_k)$ ,  $x_{k+1}^1 = \varphi_1^s(x_k^1)$ ;  $x_k^2 = \pi_{\mathbb{L}_2}(x_k)$ ,  $x_{k+1}^2 = \varphi_2^s(x_k^2)$

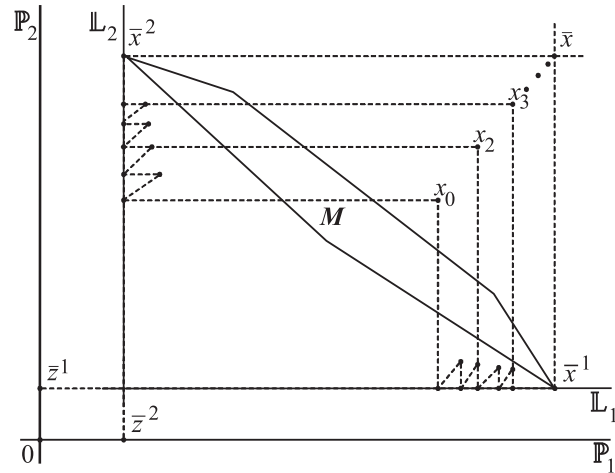


Рис. 3. Пример начального приближения  $x_0$ , дающего неверный результат

**Алгоритм  $\mathfrak{S}$ .** Пусть задано однозначное непрерывное выпуклое и замкнутое  $M$ -фейеровское отображение  $\varphi \in \{\mathbb{R}^n \rightarrow \mathbb{R}^n\}$ . Зададим разбиение пространства  $\mathbb{R}^n$  в прямую сумму ортогональных подпространств:  $\mathbb{R}^n = \mathbb{P}_1 \oplus \dots \oplus \mathbb{P}_r$ ,  $\mathbb{P}_i \perp \mathbb{P}_j$  при  $i \neq j$ . Для каждого подпространства  $\mathbb{P}_i$  ( $i = 1, \dots, r$ ) построим линейное многообразие  $\mathbb{L}_i$  следующим образом.

Пусть  $\bar{x}^i \in \text{Arg} \min_{x \in M} \rho(\mathbb{P}_i, x)$ . Положим  $\bar{z}^i = \pi_{\mathbb{P}_i^\perp}(\bar{x}^i) \in \mathbb{P}_i^\perp$ . Здесь  $\mathbb{P}_i^\perp$  обозначает ортогональное дополнение к подпространству  $\mathbb{P}_i$ . Построим линейное многообразие  $\mathbb{L}_i$  путем сдвига подпространства  $\mathbb{P}_i$  на вектор  $\bar{z}^i$ :  $\mathbb{L}_i = \mathbb{P}_i + \bar{z}^i$ . Для каждого  $i \in \{1, \dots, r\}$  определим отображение  $\varphi_i \in \{\mathbb{L}_i \rightarrow \mathbb{L}_i\}$ :

$$\varphi_i(x) = \pi_{\mathbb{L}_i}(\varphi(x)). \tag{4}$$

Зафиксируем некоторое натуральное число  $s$  и положительное вещественное число  $\varepsilon$ . Положим  $x_0 = \mathbf{0} \in \mathbb{R}^n$ . Алгоритм состоит из следующих шагов.

Шаг 0.  $k := 0$ .

Шаг 1.  $x_{k+1} := \sum_{i=1}^r (\varphi_i^s(\pi_{L_i}(x_k) - \bar{z}^i))$ .

Шаг 2.  $k := k + 1$ .

Шаг 3. Если  $\|x_{k+1} - x_k\| \geq \varepsilon$  &  $d_M(x_{k+1}) \geq \varepsilon$ , то перейти на шаг 1.

Шаг 4. Стоп.

Работа алгоритма  $\mathfrak{S}$  для размерности  $n = 2$  и  $s = 2$  схематично изображена на рис. 2.

Для применения алгоритма  $\mathfrak{S}$  к произвольной точке  $x_0 \in \mathbb{R}^n$  необходимо предварительно сделать преобразование координат, переносящее начало координат в точку  $x_0$ . Существенность этого условия иллюстрируется на рис. 3, на котором приведен пример начального приближения  $x_0$ , дающего неверный результат. После переноса начала координат в точку  $x_0$  ситуация исправляется (рис. 4).

На шаге 3 алгоритма вычисляется функция невязки  $d_M$ , которая определяет степень близости точки  $x_{k+1}$  к многограннику  $M$ . Конкретная формула для вычисления функции невязки будет приведена ниже в разделе 4.

Как будет показано в разделе 4, натуральное число  $s$  является важным параметром алгоритма  $\mathfrak{S}$ , влияющим на его масштабируемость. Увеличение  $s$  ведет к росту ресурса параллелизма, заложенного в алгоритме  $\mathfrak{S}$ . Однако при выборе слишком большого значения для параметра  $s$  последовательность точек  $\{x_k\}$ , порождаемых алгоритмом  $\mathfrak{S}$  на шаге 1, может не попасть на многогранник. В этом случае выход из итерационного процесса произойдет из-за невыполнения условия  $\|x_{k+1} - x_k\| \geq \varepsilon$ , входящего в критерий завершения. Если это произошло, то необходимо уменьшить значение  $s$  и повторить вычислительный процесс. Корректность алгоритма  $\mathfrak{S}$  обеспечивается следующей теоремой сходимости [6].

**Теорема.** Пусть  $\{x_k\}$  – последовательность точек, порождаемых алгоритмом  $\mathfrak{S}$  на шаге 1:  $x_{k+1} := \sum_{i=1}^r (\varphi_i^s(x_k) - \bar{z}^i)$ ,  $k = 0, 1, \dots$ . Тогда  $\{x_k\}_{k=0}^{+\infty} \rightarrow \bar{x} \in \mathbb{R}^n$ .

**4. Параллельная реализация алгоритма решения задачи сильной отделимости.** Алгоритм  $\mathfrak{S}$  позволяет построить высокомасштабируемую параллельную реализацию алгоритма  $\mathfrak{F}$ , решающего задачу сильной отделимости (3). В данном разделе мы опишем такую реализацию.

Сконструируем  $M$ -фейеровское отображение, следуя работе [2]. Представим систему линейных неравенств, задающих многогранник  $M$ , в следующем виде:  $Ax \leq b : \langle a_j, x \rangle - b_j \leq 0, j = 1, \dots, m$ , где  $a_j \neq 0$  для любого  $j$ . Тогда отображение вида

$$\varphi(x) = x - \sum_{j=1}^m \alpha_j \lambda_j \frac{\max\{\langle a_j, x \rangle - b_j, 0\}}{\|a_j\|^2} a_j \tag{5}$$

будет однозначным непрерывным  $M$ -фейеровским отображением для любой системы положительных коэффициентов  $\{\alpha_j > 0\}, j = 1, \dots, m$ , таких, что  $\sum_{j=1}^m \alpha_j = 1$ , и коэффициентов релаксации  $0 < \lambda_j < 2$ .

Теперь применим к (5) технику разбиения на подпространства. Пусть  $n$  – размерность пространства задачи (3). Пусть задано  $r \in \mathbb{N}: r \leq n$ . Для простоты мы будем считать, что  $r$  всегда кратно  $n: n = r \times l$ . Пусть задан ортонормированный базис пространства  $\mathbb{R}^n$ :

$$\{e_1, \dots, e_n\}. \tag{6}$$

Определим  $\mathbb{P}_i = \text{Lin}(\{e_{1+(i-1)l}, \dots, e_{l+(i-1)l}\})$  для  $i = 1, \dots, r$ . Очевидно, что  $\mathbb{P}_i \perp \mathbb{P}_j$  при  $i \neq j$  и  $\mathbb{P}_1 \oplus \dots \oplus \mathbb{P}_r = \mathbb{R}^n$ . Пусть  $\bar{x}^i \in \text{Arg} \min_{x \in M} \rho(\mathbb{P}_i, x)$ . Положим  $\bar{z}^i = \pi_{\mathbb{P}_i^\perp}(\bar{x}^i) \in \mathbb{P}_i^\perp, i = 1, \dots, r$ .

Для  $i = 1, \dots, r$  определим отображения  $\tau_i \in \{\mathbb{R}^n \rightarrow \mathbb{R}^l\}$  следующим образом. Пусть  $x \in \mathbb{R}^n, (x_1, \dots, x_n)$  – координаты вектора  $x$  в ортонормированном базисе (6). Тогда

$$\tau_i(x) = (x_{1+(i-1)l}, \dots, x_{l+(i-1)l}).$$

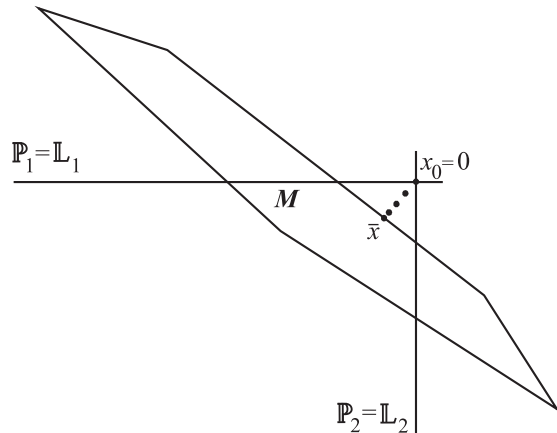


Рис. 4. Исправление ситуации после переноса начала координат в  $x_0$

В контексте формулы (5) определим  $\varphi_i \in \{\mathbb{L}_i \rightarrow \mathbb{L}_i\}$ :

$$\varphi_i(x) = \tau_i(x) - \sum_{j=1}^m \left( \alpha_j \lambda_j \frac{\max\{\langle \tau_i(a_j), \tau_i(x - \bar{z}^i) \rangle + \langle a_j, \bar{z}^i \rangle - b_j, 0\}}{\|a_j\|^2} \tau_i(a_j) \right). \quad (7)$$

Легко показать, что отображения  $\varphi_i$ , ( $i = 1, \dots, r$ ), определяемые формулой (7), удовлетворяют тождеству (4). Это обеспечивает выполнение условий алгоритма  $\mathfrak{S}$ .

Для выхода из итерационного процесса в алгоритме  $\mathfrak{S}$  на шаге 3 используется критерий завершения, включающий в себя функцию невязки  $d_M$ , определяющую степень близости точки  $x$  к многограннику  $M$ . В качестве такой функции в нашей реализации используется следующая функция:

$$d_M(x) = \sum_{j=1}^m \max\{\langle a_j, x \rangle - b_j, 0\}.$$

Аналогичным образом строится однозначное непрерывное  $N$ -фейеровское отображение  $\psi$  и набор отображений  $\psi_i$  ( $i = 1, \dots, r$ ), действующих на подпространствах.

На рис. 5 представлена схема реализации алгоритма  $\mathfrak{F}$  в виде диаграммы деятельности UML (Unified Modeling Language). Для записи операций здесь используется Си-подобный псевдокод. В массиве  $w[1..n]$  хранятся координаты текущей точки  $w_k$ , а в массиве  $w_[1..n]$  — координаты следующей точки  $w_{k+1}$ . Реализация имеет две параллельные ветви, в которых независимо выполняются шаги 1 и 2 алгоритма  $\mathfrak{F}$ . Внутри этих ветвей вычисляются псевдопроекции  $\pi_M^\varphi(w_k)$  и  $\pi_N^\psi(w_k)$  путем вызова функций  $\text{Pi\_Phi\_M}$  и  $\text{Pi\_Psi\_N}$  соответственно.

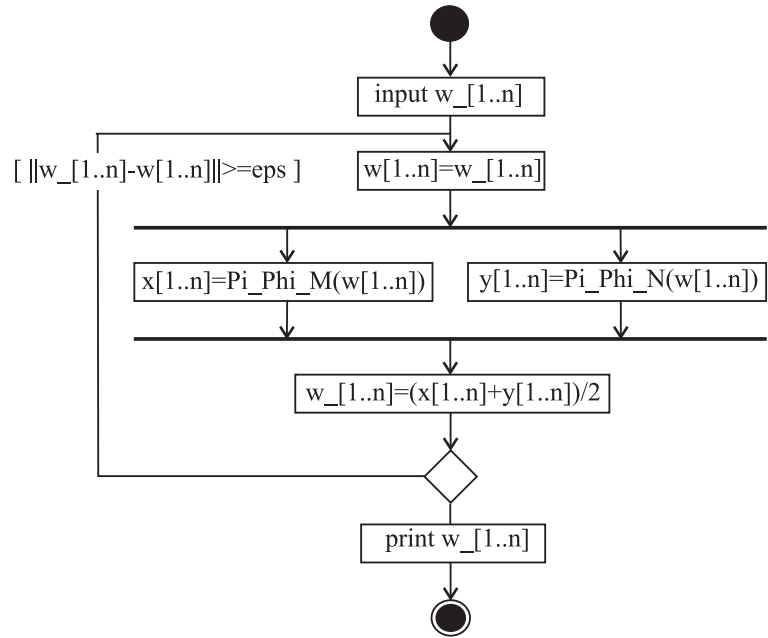


Рис. 5. Схема реализации алгоритма  $\mathfrak{F}$

Схема реализации функции  $\text{Pi\_Phi\_M}$  в виде диаграммы деятельности UML представлена на рис. 6 (реализация функции  $\text{Pi\_Psi\_N}$  строится аналогичным образом). В основу реализации  $\text{Pi\_Phi\_M}$  положен алгоритм  $\mathfrak{S}$  и фейеровские отображения (7) при  $\lambda_j = 1$  и  $\alpha_j = 1/m, j = 1, \dots, m$ . При указанных значениях отображение (7) приобретает вид

$$\varphi_i(x) = \tau_i^{-1} \left( \tau_i(x) - \sum_{j=1}^m \left( \frac{\max\{\langle \tau_i(a_j), \tau_i(x - \bar{z}^i) \rangle + \langle a_j, \bar{z}^i \rangle - b_j, 0\}}{m\|a_j\|^2} \tau_i(a_j) \right) \right). \quad (8)$$

Семантика переменных и массивов, использованных в реализации функции  $\text{Pi\_Phi\_M}$ , приведена в табл. 1. Реализация содержит  $r$  параллельных ветвей, каждая из которых вычисляет “в своем подпространстве” часть вектора  $x_k$ , применяя к нему отображение  $\varphi_i$  ( $i$  — номер подпространства)  $s$  раз, после чего происходит объединение этих частей в единый вектор. Каждая из этих параллельных ветвей может выполняться в виде независимого процесса. Это позволяет теоретически достигнуть ускорения в  $r$  раз. Действительно, циклы 1 и 2 (см. рис. 6) имеют временную сложность  $O(mn)$  и  $O(mnr)$  соответственно. Цикл 3 имеет временную сложность  $O(ml) = O(mn/r)$ . Соответственно, объемлющий его цикл 4 имеет временную сложность  $O(mns/r)$ . Цикл 5 имеет временную сложность  $O(nr)$ . Обозначим через  $h$  количество итераций цикла 6 при  $s = 1$ . Мы вправе ожидать, что количество итераций цикла 6 при  $s > 1$  будет равно  $h/s$  (в разделе 5 будут приведены результаты вычислительных экспериментов, подтверждающие это предположение). Тогда цикл 6 имеет временную сложность  $O(mnh/r + nrh/s)$ . Возьмем  $r < m$  и  $s = r$ . При этих значениях временная сложность цикла 6 принимает вид  $O(mnh/r + nrh/r) = O\left(\frac{\max\{mnh, rnh\}}{r}\right) = O(mnh/r)$ . Соответственно, временная сложность параллельной реализации, изображенной на рис. 6, составит  $O(mnh/r)$ . В то же время, временная сложность

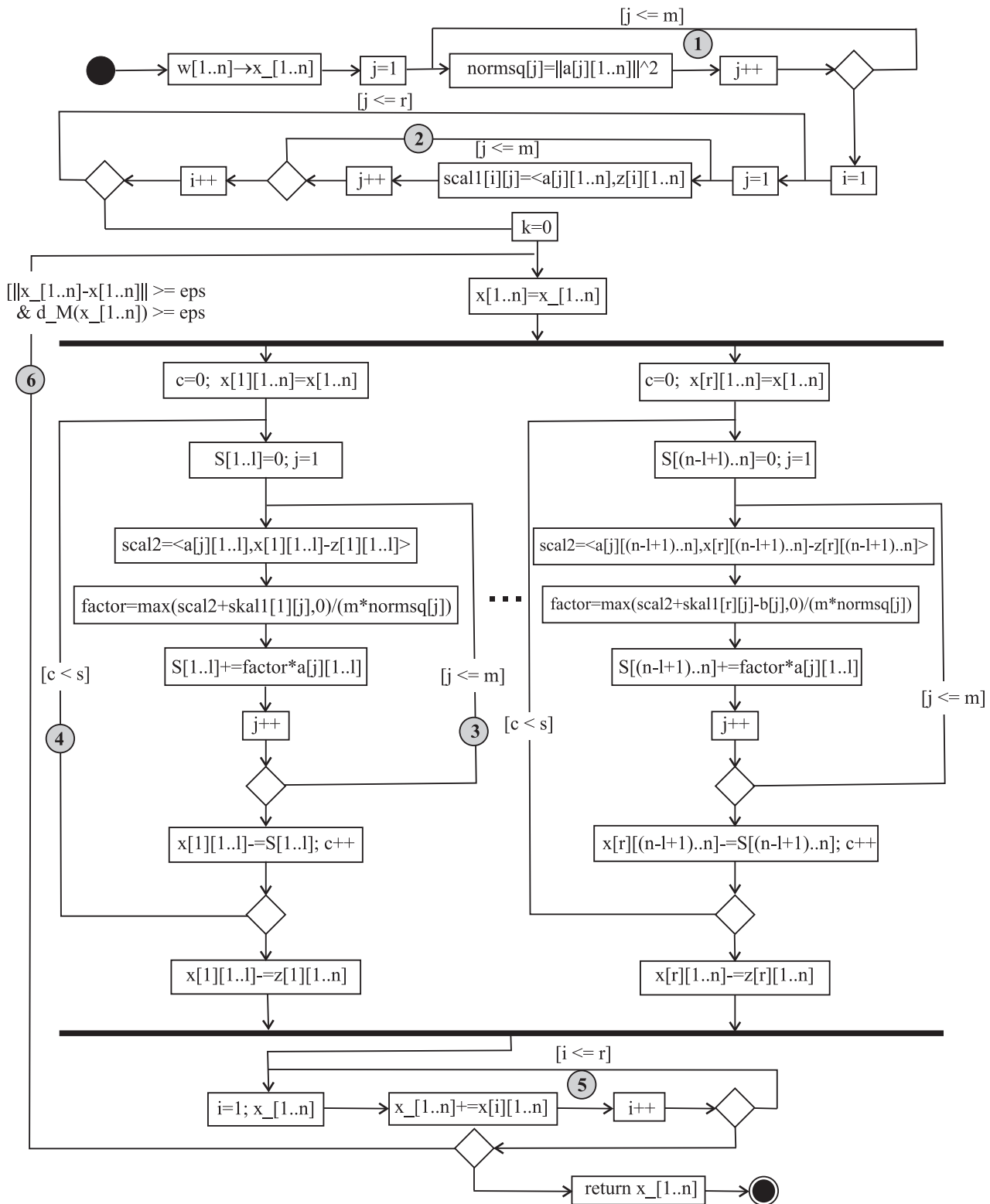


Рис. 6. Схема реализации функции Pi\_Phi\_M

последовательной реализации алгоритма вычисления псевдопроекции с помощью формулы (5) составляет  $O(mnh)$ .

На основе описанного подхода на языке программирования C++ нами была разработана параллельная программа, решающая задачу сильной отделимости многогранников для произвольных входных данных. Для организации обменов данными между процессами была использована система параллельного

Таблица 1

Семантика переменных и массивов в реализации функции Pi-Phi-M

Переменная/массив	Математическое обозначение	Семантика
n	$n$	Размерность пространства
a[j][1..n]	$a_j$	$j$ -я строка матрицы $A$
normsq[j]	$\ a_j\ ^2$	Квадрат нормы
z[i][1..n]	$\bar{z}^i$	Вектор сдвига для $\mathbb{P}_i$
scal1[i][j]	$\langle a_j, \bar{z}^i \rangle$	Скалярное произведение
x[1..n]	$x_k$	Текущее приближение к псевдопроекции на $M$
x_[1..n]	$x_{k+1}$	Следующее приближение к псевдопроекции на $M$
c	—	Счетчик независимых итераций, вычисляющих локальную переменную независимого процесса $\varphi_i^s(x_k) = \underbrace{\varphi_i \dots \varphi_i}_{s}(x_k)$
s	$s$	Количество независимых итераций, вычисляющих параметр алгоритма $\varphi_i^s(x_k) = \underbrace{\varphi_i \dots \varphi_i}_{s}(x_k)$
r	$r$	Количество подпространств (параметр алгоритма)
l	$l$	Размерность подпространства $l = n/r$
m	$m$	Количество неравенств, задающих многогранник $M$
scal2	$\langle \tau_i(a_j), \tau_i(x - \bar{z}^i) \rangle$	Скалярное произведение в пространстве $\mathbb{R}^l$ (локальная переменная независимого процесса)
factor	$\max \left\{ \langle \tau_i(a_j), \tau_i(x - \bar{z}^i) \rangle + \langle a_j, \bar{z}^i \rangle - b_j, 0 \right\} \times \frac{1}{m \ a_j\ ^2}$	Скалярный множитель под знаком суммы в формуле (8), вычисляемый в пространстве $\mathbb{R}^l$ (локальная переменная независимого процесса)
S[(1+(i-1)*1)...(1+(i-1)*1)]	$\sum_{j=1}^m (\dots)$	Сумма векторов в формуле (8), вычисляемая в пространстве $\mathbb{R}^l$ (локальный массив независимого процесса)
d_M(x_[1..n])	$d_M(x_{k+1}) = \sum_{j=1}^m \max \{ \langle a_j, x_{k+1} \rangle - b_j, 0 \}$	Невязка, определяющая меру близости точки $x_{k+1}$ к многограннику $M$
eps	$\varepsilon$	Точность приближения

программирования MPI, основанная на передаче сообщений [9]. Исходные тексты программ свободно доступны в Интернет по адресу <http://life.susu.ru/dscr/>.

Параллельная структура программы организована по иерархическому принципу. Во главе иерархии стоит процесс-“мастер”, координирующий работу остальных процессов по выполнению алгоритма  $\mathfrak{F}$ . Схе-

ма его реализации была приведена на рис. 5. Процессу “мастер” подчиняются два независимых процесса-“бригадир”. Первый процесс-“бригадир” вычисляет функцию  $Pi\_Phi\_M$ , реализация которой была приведена на рис. 6. Вторым процесс-“бригадир” вычисляет функцию  $Pi\_Psi\_N$ , реализация которой аналогична реализации функции  $Pi\_Phi\_M$ . Каждому процессу-“бригадиру” подчинено по  $r$  независимых процессов-“рабочих”. “Бригадир” делит полученный от “мастера” вектор на подвекторы и передает их “рабочим”. Каждый “рабочий” выполняет  $s$  фейеровских итераций в своем подпространстве и передает полученный подвектор “бригадиру”, который объединяет подвекторы, пришедшие от различных “рабочих”, в единый вектор. “Бригадир” проверяет критерий завершения итерационного процесса. Если он не выполнен, то “бригадир” снова делит вектор на подвекторы и передает их “рабочим”. Если критерий завершения имеет место, то полученный вектор принимается в качестве приближения псевдопроекции на многогранник и передается “мастеру”. “Мастер”, получив обе псевдопроекции от “бригадиров”, считает очередную среднюю точку и проверяет для нее критерий завершения. Если он не выполнен, то мастер передает среднюю точку каждому из “бригадиров” для продолжения вычислений.

Все указанные процессы оформляются как процессы MPI, которые могут запускаться на любом количестве процессоров (процессорных ядер), не превосходящем  $(2r + 3)$ . В предельном случае, когда каждое подпространство имеет размерность 1, имеем  $r = n$ . Иными словами, максимальная масштабируемость алгоритма равняется  $(2r + 3)$  процессоров, где  $n$  — размерность задачи.

Важным параметром реализации является количество независимых итераций  $s$ , выполняемых процессом-“рабочим”. Увеличивая  $s$ , мы можем повышать вычислительную нагрузку на те процессоры, на которых выполняются процессы-“рабочие”. Это позволяет оптимизировать затраты на пересылку сообщений между “рабочими” и “бригадирами”. Влияние параметра  $s$  на эффективность параллельного алгоритма будет исследована в следующем разделе.

**5. Вычислительные эксперименты.** Для проведения экспериментов был использован вычислительный кластер “СКИФ-Урал” с характеристиками, представленными в табл. 2.

Таблица 2

Характеристики вычислительного кластера “СКИФ-Урал”

Количество узлов/процессоров/ядер		166/332/1328
Конфигурация узла	Процессоры	2 × Intel Xeon E5472 (4 ядра по 3.0 GHz)
	RAM	8 GB
	HDD	120 GB
Тип системной сети		InfiniBand (20 Gbit/s, макс. задержка 2 мкс)
Тип управляющей (вспомогательной) сети		Gigabit Ethernet
Операционная система		SUSE Linux Enterprise Server 10
MPI-2		MVAPICH2
Компилятор		Intel C/C++ Compiler 9.0 EMT 64

**5.1. Масштабируемая модельная задача *Model-n*.** При проведении вычислительных экспериментов мы использовали сконструированную нами *модельную задачу Model-n*. Для такого типа задач можно легко аналитически вычислить точное значение толщины максимального разделяющего слоя, поэтому они хорошо подходят для проверки корректности алгоритма и изучения его масштабируемости. Кроме того, модельные задачи дают хорошую возможность для подбора оптимальных значений параметров алгоритма. Задача *Model-n* имеет вид ( $n$  задает размерность задачи), представленный в табл. 3.

Для всех размерностей  $n$  толщина максимального разделяющего слоя равна 10 000. На рис. 7 изображены многогранники  $M$  и  $N$ , задаваемые задачей *Model-3* ( $n = 3$ ), и итерации алгоритма  $\mathfrak{F}$ , приводящие к циклу неподвижности.

**5.2. Исследование параллельной реализации алгоритма  $\mathfrak{G}$ .** В первой серии вычислительных экспериментов мы исследовали количество итераций  $K$  внешнего цикла  $\mathfrak{G}$  (см. рис. 6) при вычислении псевдопроекции на многогранник  $M$  в зависимости от значений параметра  $s$ , задающего количество независимых фейеровских итераций, выполняемых процессом-“рабочим” на своем подвекторе. Эксперименты проводились при фиксированной размерности задачи  $n = 1536$  и количестве процессорных ядер  $p = 256$ . Количество подвекторов  $r$  было равно  $p$ . Значения  $s$  варьировались от 1 до 10 000. Результаты экспериментов показывают (табл. 4), что  $K \approx h/s$ , где  $h$  — количество итераций цикла  $\mathfrak{G}$  при  $s = 1$ . Среднеквадратичное отклонение значений  $K$  от значений  $h/s$  в табл. 4 составило 0.58. Среднеквадратичное

Таблица 3

Многогранник $M$	Многогранник $N$
$x_1 - 2x_2 \leq 0$	$x_1 - 2x_2 \leq 20\,000$
$x_1 - 2x_3 \leq 0$	$x_1 - 2x_3 \leq 20\,000$
...	...
$x_1 - 2x_n \leq 0$	$x_1 - 2x_n \leq 20\,000$
$x_1 + 2x_2 \leq 20\,000$	$x_1 + 2x_2 \leq 40\,000$
$x_1 + 2x_3 \leq 20\,000$	$x_1 + 2x_3 \leq 40\,000$
...	...
$x_1 + 2x_n \leq 20\,000$	$x_1 + 2x_n \leq 40\,000$
$-x_1 \leq 0$	$-x_1 \leq -20\,000$
$-x_2 \leq 0$	$-x_2 \leq 0$
...	...
$-x_n \leq 0$	$-x_n \leq 0$

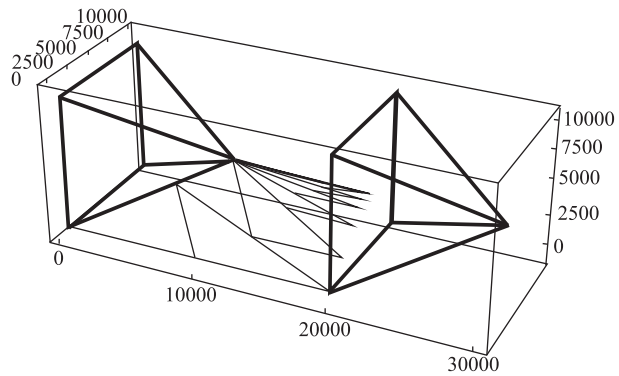


Рис. 7. Итерации алгоритма  $\mathfrak{F}$  для задачи *Model-3*

отклонение  $\sigma$  вычислялось по формуле  $\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - y_i)^2}$ , где  $x_i$  — значения из столбца “ $K$ ”,  $y_i$  —

значения из столбца “ $h/s$ ”, а  $n$  — количество строк в табл. 4. Аналогичные эксперименты были проведены для количества процессоров (и подвекторов), равного 1024 и 2048. И в том, и в другом случае вновь была получена закономерность  $K \approx h/s$  при среднеквадратичном отклонении, не превышающем 0.6. Таким образом, эксперименты подтверждают предположение о том, что  $K \approx h/s$ , сделанное в разделе 4 при оценке временной сложности параллельной реализации операции вычисления псевдопроекции.

Таблица 4  
Зависимость количества итераций  $K$  внешнего цикла от параметра  $s$

$s$	$K$	$h/s$	$s$	$K$	$h/s$	$s$	$K$	$h/s$
1	121 844	121 844	3500	35	34.8	7000	18	17.4
500	244	243.7	4000	31	30.5	7500	17	16.2
1000	122	121.8	4500	28	27.1	8000	16	15.2
1500	82	81.2	5000	25	24.4	8500	15	14.3
2000	61	60.9	5500	23	22.2	9000	14	13.5
2500	49	48.7	6000	21	20.3	9500	13	12.8
3000	41	40.6	6500	19	18.7	10 000	13	12.2

Во второй серии вычислительных экспериментов варьировался процент обменов между процессорными ядрами. Нами был введен параметр  $\omega$ , определяемый как интенсивность обменов между процессорными ядрами, измеряемый в процентах:  $\omega = \begin{cases} \frac{1}{s} \times 100\%, & s > 0, \\ 0\%, & s = 0. \end{cases}$

Мы исследовали зависимость величины отклонения  $\Delta = \|\bar{x}_{100\%} - \bar{x}_\omega\|$  от интенсивности обменов  $\omega$ . Здесь  $\bar{x}_{100\%}$  — приближение псевдопроекции, которое получалось при  $\omega = 100\%$ . В этом случае  $s = 1$ , т.е. обмены между процессами-“рабочими” происходили после каждой фейеровской итерации на подвекторах. С точки зрения точности приближения этот случай эквивалентен последовательному алгоритму построения псевдопроекции без разбиения вектора на подвекторы. Соответственно,  $\bar{x}_\omega$  — приближение псевдопроекции, которое получалось при определенном значении параметра  $\omega$ . Эксперименты проводились для трех размерностей  $n = 256$ ,  $n = 512$  и  $n = 1024$  при фиксированном количестве процессорных ядер  $p = 32$  и  $r = p$ . Интенсивность обменов  $\omega$  варьировалась от 0% до 100%. При 100% обменов параллельный алгоритм производит обмены данными между процессорными ядрами на каждой итерации. При 0% — обменов данными не происходит до конца работы алгоритма. Результаты эксперимента представ-

лены на рис. 8. Графики показывают, что при увеличении процента обменов отклонение  $\Delta$  от точного решения уменьшается для всех трех размерностей. Кроме того, чем больше размерность решаемой задачи, тем меньше величина отклонения  $\Delta$ . Максимальная величина отклонения  $\Delta$  во всех случаях не превышает 7% от высоты многомерной пирамиды, образованной системой линейных неравенств задачи *Model-n*.

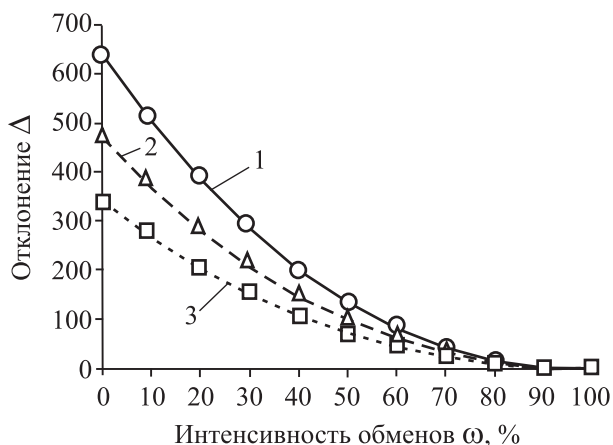


Рис. 8. Зависимость отклонения  $\Delta$  от интенсивности обменов  $\omega$  ( $r = p = 32$ ):  
1)  $n = 256$ , 2)  $n = 512$ , 3)  $n = 1024$

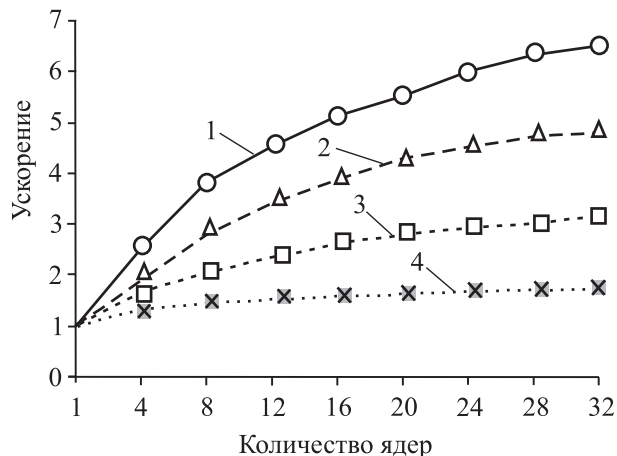


Рис. 9. Кривые ускорения при разной интенсивности обменов  $\omega$  ( $n = 512, r = p$ ):  
1)  $w = 0\%$ , 2)  $w = 30\%$ , 3)  $w = 60\%$ , 4)  $w = 100\%$

В третьей серии экспериментов исследовалось ускорение параллельной реализации алгоритма  $\mathfrak{S}$  при различной интенсивности обменов  $\omega$ . Задача решалась для четырех значений параметра  $\omega$ : 0%, 30%, 60% и 100%, при фиксированной размерности задачи  $n = 512$  и количестве подвекторов  $r = p$ . Для каждого варианта алгоритма варьировалось количество процессорных ядер, используемых для решения задачи. Результаты экспериментов приведены на рис. 9. Графики показывают, что с уменьшением процента обменов между процессорными ядрами ускорение растет, а наилучшее ускорение достигается при работе варианта алгоритма, вообще не использующего обмены. Наименьший ресурс параллелизма демонстрирует вариант параллельного алгоритма, в котором обмены производятся после каждой фейеровской итерации.

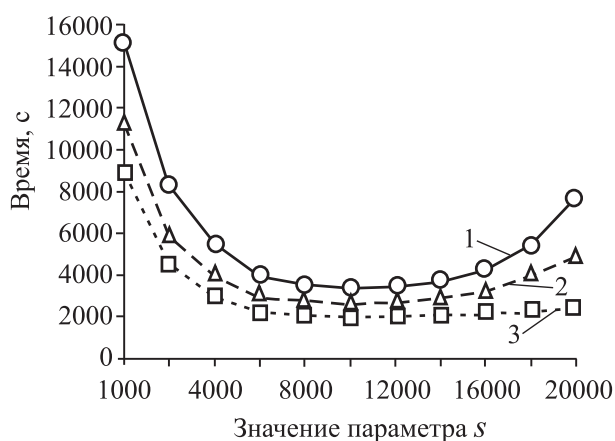


Рис. 10. Зависимость времени решения задачи от параметра  $s$  ( $r = p, n = 1536$ ): 1)  $p = 128$ ,  
2)  $p = 256$ , 3)  $p = 512$

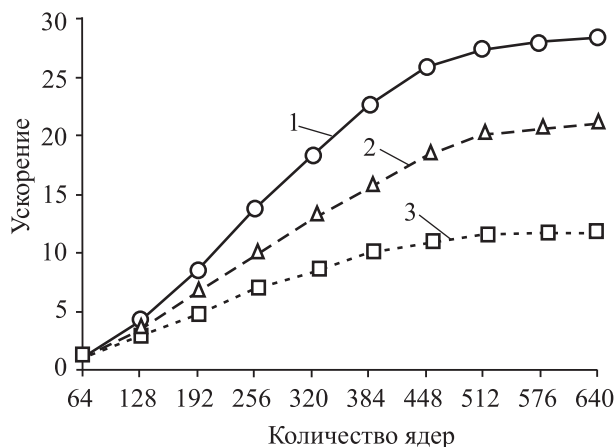


Рис. 11. Ускорение для задачи *Model-n* ( $s = 10000, r = p$ ): 1)  $n = 2048$ , 2)  $n = 1536$ , 3)  $n = 1024$

**5.3. Исследование параллельной реализации алгоритма  $\mathfrak{Z}$ .** В первой серии экспериментов исследовалась зависимость времени решения задачи *Model-n* от параметра  $s$ . Эксперименты проводились при фиксированной размерности задачи  $n = 1536$ . Значения  $s$  варьировались от 1000 до 20000. Вычисления производились на следующем количестве процессорных ядер:  $p = 128, 256$  и  $512$ . При этом количество подвекторов  $r$  также всегда было равно количеству процессорных ядер  $p$ . Результаты экспериментов показывают (рис. 10), что при увеличении значения  $s$  от 1000 до 10000 время решения задачи умень-

шается более чем на порядок. Это происходит вследствие того, что с ростом  $s$  уменьшается количество межпроцессорных обменов. Однако при дальнейшем увеличении значения  $s$  время счета также начинает увеличиваться. Это связано с тем, что при  $s$ , равном примерно 11 000, мы попадаем на многогранник уже после первой итерации цикла 6 на рис. 6. Дальнейшее увеличение количества итераций цикла 4 оказывается неэффективным, так как мы уже достигли точки неподвижности, и счет происходит “вхолостую”. Таким образом, в данном случае целесообразно выбрать значение  $s = 10\,000$ .

Во второй серии экспериментов (рис. 11) исследовалось ускорение параллельного алгоритма. Ускорение вычислялось по формуле  $\alpha = t_p/t_{64}$ , где  $t_{64}$  — время, затраченное на решение задачи *Model-n* на 64 процессорных ядрах,  $t_p$  — время, затраченное на решение этой же задачи на  $p$  процессорных ядрах. Эксперименты проводились для трех размерностей  $n = 1024$ ,  $n = 1536$  и  $n = 2048$ , при фиксированном  $s = 10\,000$  и  $r = p$ . Для каждой размерности варьировалось количество процессорных ядер, используемых для решения задачи. Во всех трех случаях мы наблюдали ускорение, близкое к линейному, вплоть до 512 процессорных ядер. Далее рост ускорения замедлялся. Однако при больших размерностях “загоризонтальвание” кривой ускорения происходило позже.

**6. Заключение.** В настоящей статье рассмотрен итерационный алгоритм  $\mathfrak{F}$  для решения задач разделения двух выпуклых непересекающихся многогранников слоем наибольшей толщины. Этот алгоритм базируется на использовании фейеровских отображений в качестве аналога операции проектирования. Предложена параллельная реализация алгоритма  $\mathfrak{F}$ , которая может быть использована без каких-либо изменений как на многопроцессорных системах с общей памятью, так и на системах с массовым параллелизмом, включая кластерные вычислительные системы. В основе алгоритма  $\mathfrak{F}$  лежит итерационный алгоритм  $\mathfrak{S}$  нахождения псевдопроекции точки на многогранник с помощью фейеровских отображений. На высокопроизводительном вычислительном кластере проведены численные эксперименты по исследованию параллельной реализации итерационного алгоритма  $\mathfrak{S}$  с использованием масштабируемых модельных задач *Model-n*. Кроме того, исследована параллельная реализация алгоритма  $\mathfrak{F}$ . Результаты экспериментов показывают, что на задачах большой размерности алгоритм  $\mathfrak{F}$  допускает эффективное масштабирование до тысячи процессоров.

В качестве возможных направлений дальнейших исследований представляются интересными следующие задачи: 1) тестирование разработанного программного комплекса на реальных задачах большой размерности; 2) разработка и исследование метода решения задачи разделения двух выпуклых непересекающихся многогранников на основе гибридной технологии, сочетающей использование фейеровских процессов и операций проектирования.

#### СПИСОК ЛИТЕРАТУРЫ

1. Боровкова В.А. Рынок ценных бумаг. СПб.: Питер, 2005.
2. Еремин И.И. Теория линейной оптимизации. Екатеринбург: Изд-во “Екатеринбург”, 1999.
3. Еремин И.И. Фейеровские методы сильной отделимости выпуклых полиэдральных множеств // Известия вузов. Сер. матем. 2006. № 12. 33–43.
4. Еремин И.И., Мазуров Вл.Д. Нестационарные процессы математического программирования. М.: Наука, 1979.
5. Ершова А.В. Алгоритм разделения двух выпуклых непересекающихся многогранников с использованием фейеровских отображений // Системы управления и информационные технологии. 2009. № 1. 53–56.
6. Ершова А.В., Соколинская И.М. О сходимости масштабируемого алгоритма построения псевдопроекции на выпуклое замкнутое множество // Вестник ЮУрГУ. Сер. “Матем. моделирование и программирование”. 2011. № 37(254), вып.10. 12–21.
7. Майоров С.И. Алгоритмическая торговля — за и против // Биржевое обозрение. 2010. № 1(73). 9–18.
8. Boser V., Guyon I., Vapnik V. A training algorithm for optimal margin classifiers // Proc. of the 5th Annual ACM Workshop on Computational Learning Theory. Pittsburgh: ACM Press, 1992. 144–152.
9. Dongarra J.J., Otto S.W., Snir M., Walker D. A message passing standard for MPP and workstations // Communications of the ACM. 1996. **39**, N 7. 84–90.
10. Forrest J.J.H., Tomlin J.A. Implementing the simplex method for the optimization subroutine library // IBM Systems Journal. 1992. **31**, N 1. 11–25.
11. Lampert A., Dale R., Paris C. Segmenting email message text into zones // Proc. of the 2009 Conference on Empirical Methods in Natural Language Processing: ACL and AFNLP. Singapore: World Scientific, 2009. 919–928.
12. Markowitz H. Portfolio selection // J. of Finance. 1952. **7**, N 1. 77–91.
13. Zhang L., Zhu J., Yao T. An evaluation of statistical spam filtering techniques // Transactions on Asian Language Information Processing. 2004. **3**, N 4. 243–269.

Поступила в редакцию  
15.10.2011