

ЧЕЛЯБИНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

На правах рукописи

СОКОЛИНСКАЯ Ирина Михайловна

**СИНТЕЗ МЕТОДОВ ОПТИМИЗАЦИИ  
И ДИСКРИМИНАНТНОГО АНАЛИЗА В  
МАТЕМАТИЧЕСКИХ МОДЕЛЯХ ЭКОНОМИКИ**

Специальность 05.13.18 - математическое моделирование,  
численные методы и комплексы программ

Диссертация на соискание ученой степени  
кандидата физико-математических наук

Научный руководитель:  
МАЗУРОВ Владимир Данилович,  
доктор физ.-мат. наук, профессор

Челябинск - 2006

## ОГЛАВЛЕНИЕ

Введение .....	4
Глава 1. Синтез дискриминантного анализа и линейной оптимизации.....	10
1.1. Математическая модель задачи ЛПНО .....	12
1.2. Общий метод решения задачи ЛПНО .....	13
1.3. Устойчивость задачи ЛПНО.....	15
1.4. Алгоритм $\mathcal{L}$ порождения образцов.....	16
1.5. Теорема сходимости для алгоритма $\mathcal{L}$ .....	20
1.6. Реализационные аспекты алгоритма $\mathcal{L}$ .....	30
1.7. Случай нескольких неформализованных ограничений.....	32
Глава 2. Алгоритм ЛП-ДА .....	36
2.1. Общая схема алгоритма ЛП-ДА.....	36
2.2. Формирование начального набора образцов .....	39
2.3. Критерий завершения итерационного процесса .....	41
2.4. Рандомизация .....	42
2.5. Метод осцилляций.....	44
2.6. Проблема погрешности вычислений .....	45
Глава 3. Программный комплекс ЛП-ДА.....	49
3.1. Модульная структура комплекса .....	49
3.2. Формат входных данных LPNC .....	51
3.3. Параллельная реализация .....	54
3.3.1. Классификация параллельных методов решения задачи ЛПНО ..	54
3.3.2. Параллельная версия алгоритма ЛП-ДА.....	56
3.4. Реализация прототипа .....	60
Глава 4. Компьютерный анализ алгоритма ЛП-ДА .....	62
4.1. Эксперименты на искусственных задачах .....	62
4.1.1. Модельная задача <i>Mod-n</i> .....	62
4.1.2. Влияние радиуса рандомизации на эффективность .....	63

4.1.3. Влияние мощности рандомизации на эффективность .....	65
4.1.4. Эффективность метода осцилляций .....	66
4.1.5. Эксперименты с неполными наборами образцов .....	67
4.2. Эксперименты на реальной задаче .....	69
4.3. Масштабируемость параллельного алгоритма .....	72
Заключение .....	78
Литература .....	83
Приложение. Основные обозначения .....	91

## ВВЕДЕНИЕ

### АКТУАЛЬНОСТЬ ТЕМЫ

В практике экономико-математического моделирования часто встречаются задачи, при решении которых приходится учитывать большое число взаимосвязанных факторов, включая плохо формализуемые [16, 21, 22, 25]. Процесс решения плохо формализуемой задачи включает в себя преобразование ее формулировки путем уточнений и упрощений. В результате этого процесса мы получаем формализованную задачу, имеющую некоторое отношение к исходной постановке задачи. Во многих случаях удается построить формализованную задачу в виде задачи линейного программирования [7, 12, 14, 18, 52]. Однако решение формализованной задачи может сколь угодно сильно отличаться от решения исходной задачи в силу наличия неформализованных ограничений.

Помимо формальных методов и моделей для решения плохо формализуемой задачи используются средства неформального анализа, включающие в себя суждения экспертов для учета так называемых «внемодельных» факторов [26, 51]. При этом человеческий компонент в процессе экспертизы может постепенно, с помощью обучения, заменяться машинным компонентом путем использования экспертных систем [8, 36, 54] или нейросетевых программ [28, 29, 30, 63, 64]. Однако при решении задач линейного программирования большой размерности эти подходы могут оказаться неэффективными, если их применять для решения всей задачи в целом. Это объясняется тем, что трудно адекватно настроить (обучить) экспертную систему или нейронную сеть в случае, когда система ограничений содержит тысячи линейных неравенств и десятки тысяч переменных. В таких ситуациях перспективным является подход, основанный на сочетании методов линейной оптимизации с процедурами экспертного оценивания. Алгоритм решения задачи в этом случае строится как итерационный процесс, в

ходе которого шаг за шагом происходит уточнение ограничений, относящихся к неформализованной части задачи. Приближенные решения исходной задачи, получаемые в ходе итерационного процесса, формируют множество образцов, квалифицируемых экспертом. Применяя к этому множеству образцов процедуру дискриминантного анализа [23, 32, 37, 66, 71], мы получаем разделяющую поверхность, аппроксимирующую неформализованные ограничения.

В соответствии с этим *актуальной* является задача разработки, анализа и реализации на ЭВМ алгоритмов решения задач линейного программирования с неформализованными ограничениями путем синтеза методов дискриминантного анализа и линейной оптимизации.

## **ЦЕЛЬ И ЗАДАЧИ ИССЛЕДОВАНИЯ**

*Цель* данной работы состояла в разработке и исследовании методов и алгоритмов решения задач линейного программирования с неформализованными ограничениями и их реализации в виде программного комплекса. Для достижения этой цели необходимо было решить следующие задачи:

1. Построить математическую модель задачи *линейного программирования с неформализованными ограничениями* (ЛПНО) и описать общий подход к ее решению.
2. На базе данного подхода разработать метод решения задачи ЛПНО, допускающий реализацию в виде алгоритма для ЭВМ, и исследовать сходимость этого метода.
3. На основе описанного метода разработать и исследовать алгоритм ЛП-ДА (Линейное Программирование – Дискриминантный Анализ), соединяющий алгоритмы линейного программирования с алгоритмами дискриминантного анализа.

4. Спроектировать и реализовать программный комплекс для решения задач ЛПНО, использующий предложенные методы и алгоритмы.
5. Провести вычислительные эксперименты для анализа эффективности предложенного подхода.
6. Разработать и реализовать параллельную версию алгоритма ЛП-ДА. Провести вычислительные эксперименты на кластерной системе для исследования ускорения и расширяемости параллельного алгоритма.

### **МЕТОДЫ ИССЛЕДОВАНИЯ**

В исследованиях, проводимых в диссертационной работе, используется аппарат теории математического программирования и распознавания образов, применяются методы дискриминантного анализа и линейной оптимизации.

### **НАУЧНАЯ НОВИЗНА**

Научная новизна работы заключается в следующем:

1. предложен метод решения задачи линейного программирования с неформализованными ограничениями, базирующийся на синтезе методов дискриминантного анализа и методов линейной оптимизации, и доказана сходимость генерируемой им итерационной последовательности к точному решению задачи ЛПНО;
2. предложен оригинальный метод осцилляций, позволяющий существенно ускорить сходимость последовательности приближений к точному решению задачи ЛПНО;
3. разработан новый алгоритм, соединяющий симплекс-метод и метод линейной коррекции, и предложена его параллельная версия;

4. выполнена реализация алгоритма решения задач ЛПНО в виде программного комплекса для персональных компьютеров и многопроцессорных систем на основе стандарта MPI-2.

## **ТЕОРЕТИЧЕСКАЯ И ПРАКТИЧЕСКАЯ ЦЕННОСТЬ**

*Теоретическая ценность* работы состоит в том, что в ней представлены доказательства теоремы об устойчивости задачи линейного программирования по неформализованному ограничению и теоремы о сходимости метода, соединяющего дискриминантный анализ и рандомизацию с линейной оптимизацией. *Практическая ценность* работы заключается в том, что предложенный программный комплекс в сочетании с системами экспертного оценивания может быть использован для решения плохо формализуемых задач, возникающих в экономико-математическом моделировании.

## **СТРУКТУРА И ОБЪЕМ РАБОТЫ**

Диссертация состоит из введения, четырех глав, заключения, библиографии и приложения, в котором приводятся основные обозначения, используемые в диссертационной работе. Объем диссертации составляет 92 страницы, объем библиографии – 77 наименований.

## **СОДЕРЖАНИЕ РАБОТЫ**

**Первая глава, «Синтез дискриминантного анализа и линейной оптимизации»**, посвящена описанию и исследованию общего метода решения задач ЛПНО, сочетающего дискриминантный анализ, линейную оптимизацию и рандомизацию. Строится общая математическая модель задачи и процесса нахождения ее приближенного решения. Доказывается теорема об устойчивости задачи линейного программирования по неформализованному ограничению. Эта теорема играет важную роль при решении проблемы сходимости. Далее описывается алгоритм получения новых образцов, служащих основой итерационного процесса. Центральной частью

главы является доказательство сходимости полученной последовательности приближенных решений к точному решению исходной задачи. Также рассматриваются вопросы применимости предложенного алгоритма для решения практических задач. В заключительном разделе первой главы показывается, как полученные результаты могут быть распространены на случай произвольного количества неформализованных ограничений.

**Во второй главе, «Алгоритм ЛП-ДА»,** предлагается вариант реализации метода, описанного в первой главе, в виде алгоритма, основанного на синтезе алгоритмов линейного программирования и дискриминантного анализа. Такую комбинацию мы обозначаем как *алгоритм ЛП-ДА*. Рассматриваются различные аспекты реализации алгоритма ЛП-ДА в виде программы для ЭВМ. Предлагается метод формирования начального набора образцов. Вводится и математически обосновывается критерий завершения итерационного процесса. Дается формальное описание процедуры рандомизации и вводятся такие параметры алгоритма, как радиус и мощность рандомизации. Описывается оригинальный метод осцилляций, позволяющий во многих случаях значительно повысить эффективность алгоритма ЛП-ДА. Предлагается система допусков, обеспечивающая корректную работу алгоритма при возникновении погрешностей вычислений, связанных с неточным представлением вещественных чисел в ячейках памяти ЭВМ.

**В третьей главе, «Программный комплекс ЛП-ДА»,** рассматривается архитектура программного комплекса для решения задач линейного программирования с неформализованными ограничениями на базе использования алгоритма ЛП-ДА. Предлагается модульная структура программного комплекса, использующая технологию картриджей, которая позволяет расширять программный комплекс путем добавления новых методов линейной оптимизации и дискриминантного анализа. Описывается оригинальный формат входных данных LPNC для представления задач ЛПНО,

базирующийся на стандарте MPS. Рассматривается параллельная реализация комплекса для многопроцессорных систем на основе стандарта MPI-2.

**В четвертой главе, «Компьютерный анализ алгоритма ЛП-ДА»,** приводится описание реальных и искусственных задач, использованных для проверки эффективности предложенных подходов, методов и алгоритмов. Описываются и обсуждаются результаты экспериментов, проведенных на персональном компьютере и на высокопроизводительном вычислительном кластере. Исследуется масштабируемость параллельной версии алгоритма ЛП-ДА для варианта, использующего симплекс-метод и метод линейной коррекции.

**В заключении** суммируются основные результаты диссертационной работы, выносимые на защиту, приводятся данные о публикациях и апробациях автора по теме диссертации, и рассматриваются направления дальнейших исследований в данной области.

**В приложении** приводятся основные сокращения и обозначения, используемые в диссертационной работе.

## ГЛАВА 1. СИНТЕЗ ДИСКРИМИНАНТНОГО АНАЛИЗА И ЛИНЕЙНОЙ ОПТИМИЗАЦИИ

Учет неформализованных ограничений в задачах экономико-математического моделирования может быть осуществлен путем сочетания оптимизационных моделей с процедурами экспертного оценивания. В основе последних лежит накопленный опыт, описываемый большими массивами плохо структурированной информации. Это составляет естественную основу для применения методов распознавания образов [22], позволяющих вносить упорядочение в такие массивы в соответствии с теми или иными факторами.

*Задача распознавания образов* [6, 16, 37, 55] заключается в нахождении целесообразного разбиения заданной совокупности объектов на классы эквивалентности. При этом различают *задачу дискриминантного анализа* (построение поверхностей, разделяющих конечные множества) и *задачу таксономии* (разбиение конечного множества на подмножества близких друг к другу элементов). *Задача дискриминантного анализа* [23] сводится к решению относительно  $\psi \in F \subset \{\mathbb{R}^n \rightarrow \mathbb{R}^1\}$  системы

$$\begin{cases} \psi(x) > 0, & x \in M \\ \psi(x) < 0, & x \in N, \end{cases}$$

где  $M$  и  $N$  – конечные непересекающиеся множества из пространства  $\mathbb{R}^n$ ;  $F$  – заданный класс функций  $n$ -мерного вектора. Функция  $\psi$  в этом случае называется *разделяющей функцией* для множеств  $M$  и  $N$ .

Для решения задачи дискриминантного анализа могут быть использованы *метод линейной коррекции* [15, 17, 33], метод комитетов [24, 26], методы фейеровского типа [13] и др.

Одной из наиболее важных оптимизационных моделей в экономико-математическом программировании является модель, описываемая в виде задачи линейного программирования [12]. Под *задачей линейного программирования* (ЛП) в пространстве  $\mathbb{R}^n$  понимают задачу максимизации линейного функционала  $(c, x)$  на множестве решений системы

$$(a_i, x) \leq b_i, \quad i = 1, \dots, m,$$

где  $a_i \in \mathbb{R}^n$ ,  $b_i \in \mathbb{R}$ . Переписав эту систему в матричном виде  $Ax \leq b$ , мы можем записать задачу линейного программирования следующим образом:

$$\max \{(c, x) \mid Ax \leq b\}.$$

Стандартным методом решения задач линейного программирования является *симплекс-метод* и его модификации [7, 47, 52, 62]. Однако в последнее время все большую популярность приобретают методы итерационного типа, получившие в работах И.И. Еремина название *фейеровских* [3, 9, 10]. Методы *фейеровского* типа могут эффективно применяться для решения задач линейного программирования большой размерности в условиях неполных, противоречивых и изменяющихся исходных данных. Кроме этого, они допускают эффективную реализацию на многопроцессорных системах с массовым параллелизмом [2].

В данной главе описывается общий метод решения задач линейного программирования с неформализованными ограничениями, соединяющий в себе методы дискриминантного анализа и линейной оптимизации. Сначала рассматривается случай одного неформализованного ограничения, а затем показывается, как полученные результаты могут быть обобщены на случай произвольного количества ограничений.

### 1.1. Математическая модель задачи ЛПНО

Рассмотрим сначала случай, когда только одно ограничение не может быть формализовано. Данную ситуацию можно записать в виде следующей задачи *линейного программирования с неформализованным ограничением* (ЛПНО) в пространстве  $\mathbb{R}^n$ :

$$\max \{(c, x) \mid Ax \leq b, \bar{\varphi}(x) \leq 0\} \quad (1.1)$$

Здесь неравенство  $\bar{\varphi}(x) \leq 0$  играет роль неформализованного ограничения в том смысле, что нам не известен вид функции  $\bar{\varphi}$ . Будем предполагать, что  $\bar{\varphi}$  принадлежит к классу аффинных функций. Предположим также, что задача (1.1) имеет единственное решение:

$$\bar{x} = \arg \max \{(c, x) \mid Ax \leq b, \bar{\varphi}(x) \leq 0\}.$$

Неформализованное ограничение  $\bar{\varphi}(x) \leq 0$  представляется двумя информационными ресурсами: *экспертом* и *набором образцов*. Рассмотрим каждый из этих информационных объектов.

*Эксперт* – это некоторый информационный ресурс, который мы можем использовать для работы с неформализованным ограничением. В качестве абстрактной модели эксперта введем функцию  $\epsilon(x) = \text{sgn}(\bar{\varphi}(x))$ . С помощью функции эксперта  $\epsilon$  мы можем для любого  $x \in \mathbb{R}^n$  определить, удовлетворяет  $x$  неформализованному ограничению или нет, используя следующую систему правил:

$$\begin{cases} \epsilon(x) = 1 & \Leftrightarrow & \bar{\varphi}(x) > 0; \\ \epsilon(x) = 0 & \Leftrightarrow & \bar{\varphi}(x) = 0; \\ \epsilon(x) = -1 & \Leftrightarrow & \bar{\varphi}(x) < 0. \end{cases} \quad (1.2)$$

На практике роль эксперта может играть нейронная сеть [48, 59, 64], экспертная система [8, 36, 54] или человек. В общем случае мы должны допускать существование определенных ограничений на использование эксперта.

Например, общее количество обращений к эксперту может быть ограничено, или ответы эксперта являются верными с некоторой вероятностью.

Набор образцов  $\mathfrak{P} = \{\mathbb{A}, \mathbb{B}\}$  определяется путем задания пары конечных непересекающихся множеств  $\mathbb{A}$  и  $\mathbb{B}$  из пространства  $\mathbb{R}^n$ , удовлетворяющих условиям

$$\begin{cases} \epsilon(x) = 1, & \forall x \in \mathbb{A}; \\ \epsilon(x) = -1, & \forall x \in \mathbb{B}. \end{cases} \quad (1.3)$$

Набор образцов  $\mathfrak{P}$  представляет собой знания о неформализованном ограничении, накопленные в результате предшествующего опыта. Постепенно пополняя  $\mathfrak{P}$  путем использования функции эксперта  $\epsilon$ , мы тем самым уточняем нашу модель экономической ситуации.

## 1.2. Общий метод решения задачи ЛПНО

Применяя методы дискриминантного анализа, мы можем найти для множеств  $\mathbb{A}$  и  $\mathbb{B}$  разделяющую функцию  $\tilde{\varphi} \in \Phi$  ( $\Phi$  – класс всех аффинных функций), удовлетворяющую условиям

$$\begin{cases} \tilde{\varphi}(x) > 0, & x \in \mathbb{A}; \\ \tilde{\varphi}(x) < 0, & x \in \mathbb{B} \end{cases}$$

(существование  $\tilde{\varphi}$  следует из (1.2), (1.3) и предположения, что  $\bar{\varphi} \in \Phi$ ). В качестве приближенного решения задачи ЛПНО мы можем теперь взять решение (если оно существует) следующей задачи линейного программирования, называемой *аппроксимационной* (ЛПА):

$$\tilde{x} = \arg \max \{ (c, x) \mid Ax \leq b, \tilde{\varphi}(x) \leq 0 \}.$$

При этом следует отметить, что приближенное решение  $\tilde{x}$  будет с приемлемой точностью аппроксимировать точное решение  $\bar{x}$ , если набор образцов  $\mathfrak{P}$  будет достаточно представительным.

Для практического использования описанного подхода нам необходимо решить следующие задачи.

- (1) Определить алгоритм  $\mathfrak{L}$  построения последовательности новых образцов.
- (2) Доказать, что последовательность решений  $\{\tilde{x}^k\}_{k=0}^{\infty}$  задач ЛПА, получающихся при последовательном добавлении новых образцов к набору  $\mathfrak{P}$ , сходится к точному решению  $\bar{x}$  задачи ЛПНО.

Решению этих вопросов посвящены разделы 1.4 и 1.5 данной главы.

**Введем систему обозначений и опишем ограничения**, используемые во всех следующих разделах диссертации.

Пусть  $D = \{x \mid Ax \leq b\}$  – многогранник, определяемый формализованной частью задачи ЛПНО. Будем предполагать, что  $D$  – телесное множество (то есть  $D$  содержит внутренние точки), и что  $D$  ограничен. Перепишем задачу ЛПНО в виде

$$\max \{(c, x) \mid x \in D; \bar{\varphi}(x) \leq 0\}. \quad (1.4)$$

Будем рассматривать случай, когда задача (1.4) имеет единственное решение:

$$\bar{x} = \arg \max \{(c, x) \mid x \in D; \bar{\varphi}(x) \leq 0\}.$$

Обозначим как *ЛПФ задачу линейного программирования с формализованной частью*:

$$\max \{(c, x) \mid x \in D\}. \quad (1.5)$$

Будем исходить из предположения, что задача (1.5) также имеет единственное решение:

$$\hat{x} = \arg \max \{(c, x) \mid x \in D\}$$

причем

$$\hat{x} \neq \bar{x}. \quad (1.6)$$

Пусть  $\bar{P} = \{x \mid \bar{\varphi}(x) \leq 0\}$  – множество значений, удовлетворяющих неформализованному ограничению  $\bar{\varphi}(x) \leq 0$ . Так как  $\bar{\varphi}$  является аффинной функцией, то  $\bar{P}$  представляет собой замкнутое полупространство. Допустимое множество значений задачи (1.4) будет иметь вид  $D \cap \bar{P}$ . Будем также предполагать, что  $D \cap \bar{P}$  содержит внутренние точки. Наконец, определим:  $\bar{H} = \{x \mid \bar{\varphi}(x) = 0\}$  – гиперплоскость, определяемая неформализованным ограничением.

### 1.3. Устойчивость задачи ЛПНО

Вычислительные методы решения задач линейного программирования с неформализованными ограничениями и организация самих вычислительных процессов сопряжены с трудностями, связанными с обеспечением точности получаемых результатов в ситуации неточных исходных данных. Главным фактором, позволяющим преодолеть эти трудности, является фактор устойчивости [11]. В данном разделе мы исследуем *устойчивость задачи ЛПНО по неформализованному ограничению*. Это условие имеет большое значение при исследовании вопросов сходимости, рассматриваемых в разделе 1.5.

Справедлива следующая теорема.

**Теорема 1.** Задача ЛПНО (1.1) является устойчивой по  $\bar{\varphi}$ .

*Доказательство.* Рассмотрим систему ограничений задачи (1.1):

$$\bar{A}x \leq \bar{b}, \quad x \geq 0. \quad (1.7)$$

Здесь система  $\bar{A}x \leq \bar{b}$  получается из  $Ax \leq b$  путем интегрирования в нее ограничения  $\bar{\varphi}(x) \leq 0$  (это всегда можно сделать, когда  $\bar{\varphi}$  является аффинной функцией). Так как  $D \cap \bar{P}$  является ограниченным выпуклым замкнутым многогранником, содержащим внутренние точки, то

$\exists x' \geq 0 : \bar{A}x' < \bar{b}$ . Отсюда по лемме 35.1 [12] получаем, что система (1.7) является устойчивой по  $\bar{b}$ . В соответствии с теоремой 36.2 [12] из  $\bar{b}$ -устойчивости системы (1.7) следует устойчивость задачи (1.1) по  $\bar{b}$ .

Из ограниченности допустимого множества  $D \cap \bar{P}$  задачи (1.1) следует ограниченность ее оптимального множества, откуда по теореме 36.1 [12] следует устойчивость задачи (1.1) по  $c$ .

Из устойчивости по  $\bar{b}$  и  $c$  легко получить устойчивость по  $[\bar{b}, c]$ , откуда по теореме 36.7 [12] следует устойчивость задачи (1.1) по всей совокупности данных. В частности, мы получаем устойчивость задачи (1.1) по  $\bar{\varphi}$ , что и требовалось доказать.

#### **1.4. Алгоритм $\mathfrak{L}$ порождения образцов**

Алгоритм порождения образцов является ключевым фактором практической применимости подхода, описанного в разделе 1.2. Это связано с тем, что, функция эксперта  $\epsilon$  на практике часто имеет ограничение на количество обращений. В соответствии с этим мы должны строить алгоритм порождения образцов таким образом, чтобы количество обращений к функции эксперта  $\epsilon$  было минимальным.

В общем виде предлагаемый алгоритм, обозначим его  $\mathfrak{L}$ , может быть определен следующим образом.

**Алгоритм  $\mathfrak{L}$ .** Пусть заданы конечные множества  $\mathbb{A}^0$  и  $\mathbb{B}^0$ , такие, что  $\mathbb{A}^0 \subset D \setminus \bar{P}$ ,  $\mathbb{B}^0 \subset D \cap (\bar{P} \setminus \bar{H})$ . Множества  $\mathbb{A}^0$  и  $\mathbb{B}^0$  задают начальный набор образцов  $\mathfrak{P}^0$ . Будем предполагать, что  $\mathbb{A}^0 \neq \emptyset$  и  $\mathbb{B}^0 \neq \emptyset$ . Алгоритм  $\mathfrak{L}$  состоит из следующих шагов:

Шаг 0.  $k := 0$ .

Шаг 1 (дискриминантный анализ). Найдем аффинную функцию  $\tilde{\varphi}^k$ , удовлетворяющую условиям

$$\forall x \in \mathbb{A}^k : \tilde{\varphi}^k(x) > 0, \forall x \in \mathbb{B}^k : \tilde{\varphi}^k(x) < 0. \quad (1.8)$$

и условию нормализации: вектор нормали к гиперплоскости  $\tilde{H}^k = \{x \mid \tilde{\varphi}^k(x) = 0\}$  должен быть коллинеарен вектору нормали к гиперплоскости  $\bar{H}$ .

Шаг 2 (порождение образцов). Получим новый образец как решение следующей задачи ЛПА<sup>k</sup>:

$$\tilde{x}^k = \arg \max \{(c, x) \mid x \in D, \tilde{\varphi}^k(x) \leq 0\}. \quad (1.9)$$

Шаг 3 (экспертиза и пополнение  $\mathfrak{P}$ ).

Если

$$(c, \tilde{x}^k) \notin \{(c, x) \mid x \in \mathbb{A}^k \cup \mathbb{B}^k \cup \bar{H}\}, \quad (1.10)$$

ТО ПОЛОЖИМ

$$\mathbb{A}^{k+1} = \begin{cases} \mathbb{A}^k, & \mathbf{e}(\tilde{x}^k) = -1 \\ \mathbb{A}^k \cup \{\tilde{x}^k\}, & \mathbf{e}(\tilde{x}^k) = 1 \end{cases}$$

$$\mathbb{B}^{k+1} = \begin{cases} \mathbb{B}^k \cup \{\tilde{x}^k\}, & \mathbf{e}(\tilde{x}^k) = -1 \\ \mathbb{B}^k, & \mathbf{e}(\tilde{x}^k) = 1 \end{cases}$$

Шаг 4 (рандомизация). Если условие (1.10) не выполняется, то выполним процедуру рандомизации. Пусть

$$\mathfrak{R}^\rho(C) = \{x + r_i^\rho(x) \mid \forall x \in C; i = 1, \dots, m\},$$

где  $r_i^\rho(x)$  – случайный вектор в  $\rho$ -окрестности точки  $x$ . Будем называть  $\rho$  радиусом рандомизации,  $m$  – мощностью рандомизации ( $m \geq 1$ ). Выберем  $\rho_k$  такое, что

$$0 < \rho_k < \min(\text{dist}(\mathbb{A}^k, \bar{H}), \text{dist}(\mathbb{B}^k, \bar{H})) \quad (1.11)$$

(здесь  $\text{dist}(X, Y) = \inf \{\|x - y\| : x \in X, y \in Y\}$ ).

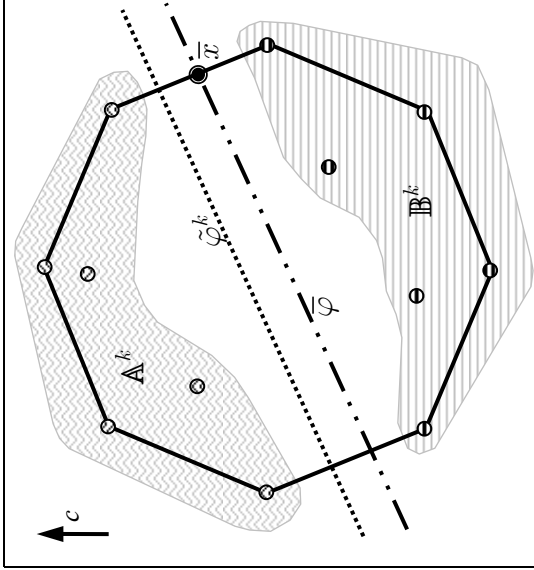
Положим:

$$\mathbb{A}^{k+1} = \mathbb{A}^k \cup \mathfrak{R}^{\rho_k}(\mathbb{A}^k), \quad \mathbb{B}^{k+1} = \mathbb{B}^k \cup \mathfrak{R}^{\rho_k}(\mathbb{B}^k).$$

Шаг 5.  $k := k + 1$ .

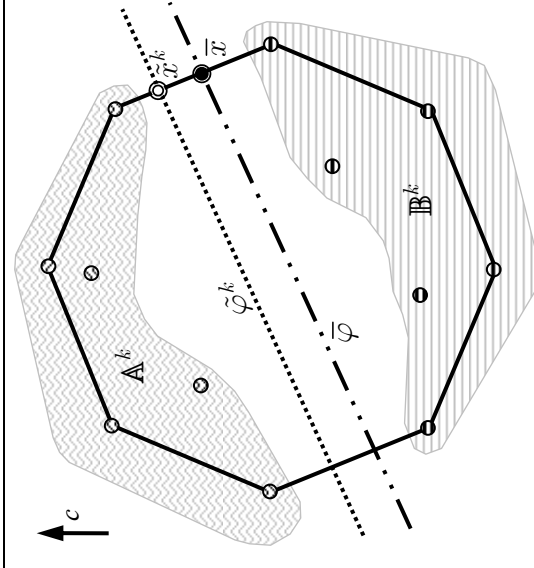
Шаг 6. Перейти на шаг 1.

Шаг 1. Нахождение разделяющей функции.

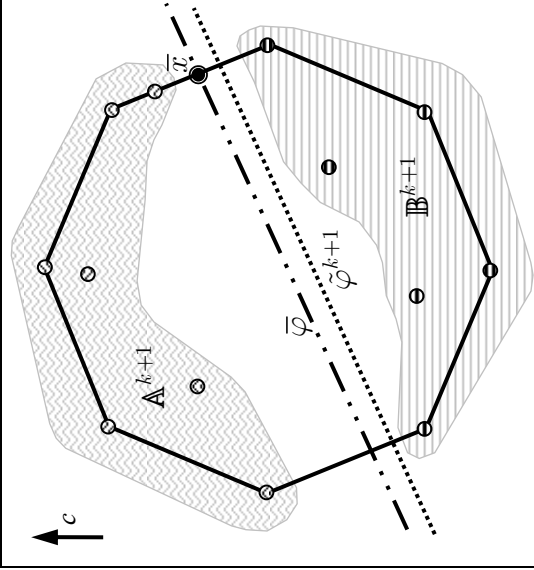
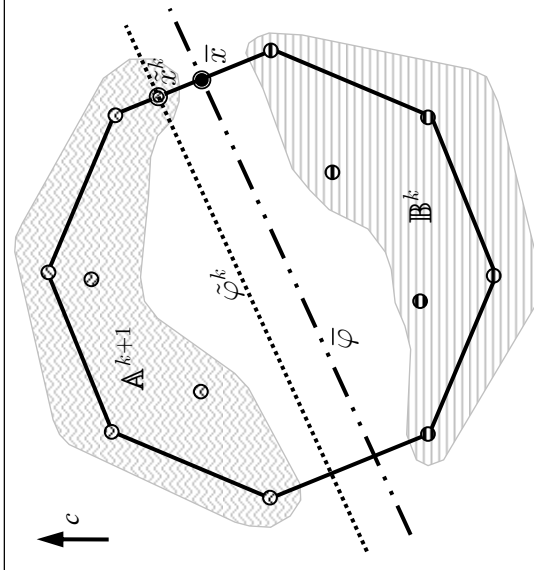


Итерация  
 $k$

Шаг 2. Порождение нового образа.



Шаг 3. Экспертиза.



Итерация  
 $k+1$

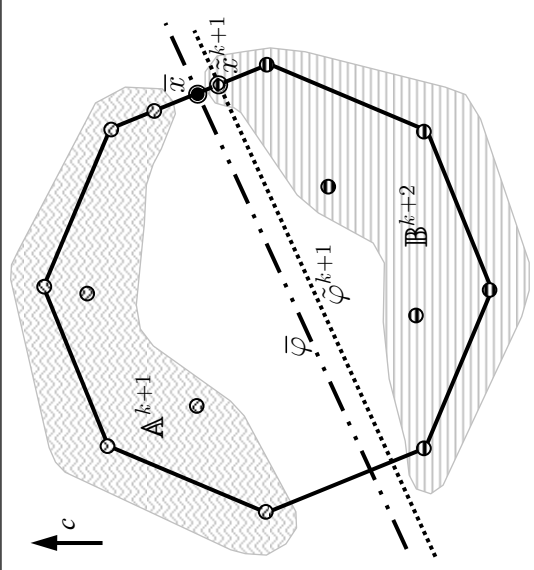
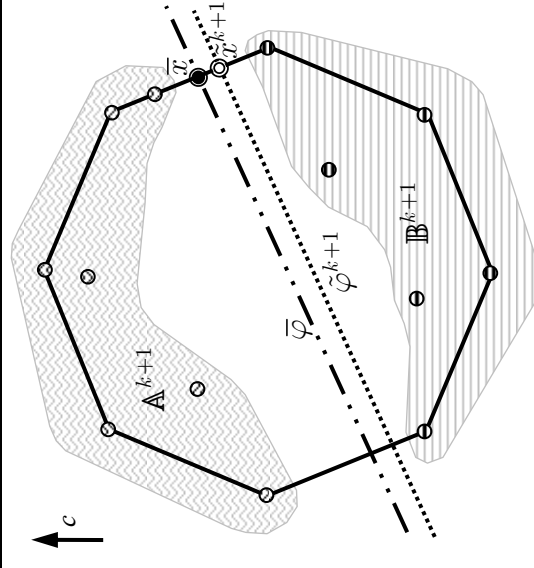


Рис. 1. Итерации алгоритма  $\Omega$ .

Схематично итерации алгоритма  $\mathfrak{L}$  изображены на Рис. 1. Применение алгоритма  $\mathfrak{L}$  для нахождения решения задачи ЛПНО (1.4) заключается в последовательном вычислении точек  $\tilde{x}^k$ , являющихся решением задач ЛПА<sup>k</sup>:

$$\max \{(c, x) \mid x \in D, \tilde{\varphi}^k(x) \leq 0\}. \quad (1.12)$$

Процесс завершается при выполнении некоторого *критерия*, указывающего на то, что очередной  $\tilde{x}^k$  оказался в достаточной близости от точного решения  $\bar{x}$ . Проблема выбора критерия завершения итерационного процесса будет рассмотрена в разделе 2.3.

Алгоритм  $\mathfrak{L}$  накладывает на разделяющую гиперплоскость  $\tilde{H}^k$ , получаемую на шаге 1, ограничение, названное условием нормализации. Это условие будет использовано при доказательстве теоремы сходимости в разделе 1.5. Условие нормализации требует, чтобы гиперплоскость  $\tilde{H}^k$ , задаваемая разделяющей функцией, была параллельна гиперплоскости  $\bar{H}$ , задаваемой неформализованным ограничением. В разделе 1.6 будет показано, что условием нормализации можно пренебречь, если в практической реализации алгоритма  $\mathfrak{L}$  регулярно выполняется процедура рандомизации.

Условие (1.11) гарантирует нам выполнение следующих основных свойств набора образцов  $\mathfrak{P}^k = \{\mathbb{A}^k, \mathbb{B}^k\}$ :  $\mathbb{A}^k \cap \bar{P} = \emptyset$ ,  $\mathbb{B}^k \subset \bar{P} \setminus \bar{H}$ . Вопрос практической реализации условия (1.11) будет рассмотрен в разделе 1.6. Там же будет рассмотрена проблема реализации условия (1.10).

В заключение отметим, что в основе алгоритма  $\mathfrak{L}$  лежит алгоритм  $\mathfrak{A}$ , описанный в [23]. В отличие от алгоритма  $\mathfrak{L}$ , в алгоритме  $\mathfrak{A}$  не применяется процедура рандомизации. Кроме этого, в качестве условия нормализации алгоритм  $\mathfrak{A}$  использует следующее условие:

$$\forall k : \text{dist}(\mathbb{A}^k, \tilde{H}^k) = \text{dist}(\mathbb{B}^k, \tilde{H}^k).$$

Для практического использования алгоритма  $\mathfrak{L}$ , предложенного в настоящем разделе, важным является вопрос сходимости последовательно-

сти  $\{\tilde{x}^k\}_{k=0}^{\infty}$  к точному решению задачи ЛПНО (1.4). Сходимость алгоритма  $\mathfrak{A}$  была исследована в [26, стр. 173]. Следующий раздел посвящен вопросу сходимости алгоритма  $\mathfrak{L}$ .

### 1.5. Теорема сходимости для алгоритма $\mathfrak{L}$

Справедлива следующая теорема.

**Теорема 2.** Для последовательности  $\tilde{x}^0, \tilde{x}^1, \dots, \tilde{x}^k$ , определяемой алгоритмом  $\mathfrak{L}$ , имеем:

$$\{\tilde{x}^k\}_{k=0}^{\infty} \rightarrow \bar{x}.$$

Доказательству теоремы 2 предположим леммы 1-3.

**Лемма 1.** Пусть для всех достаточно больших  $k$  задача ЛПА<sup>k</sup>

$$\tilde{\mu}^k = \max \{ (c, x) \mid x \in D, \tilde{\varphi}^k(x) \leq 0 \},$$

порождаемая алгоритмом  $\mathfrak{L}$ , имеет бесконечное множество решений  $\tilde{X}^k = \{x \mid x \in D \cap \tilde{P}^k; (c, x) = \tilde{\mu}^k\}$ . Тогда

$$\lim_{k \rightarrow \infty} \text{dist}(\tilde{H}^k, \bar{H}) = 0 \quad \Rightarrow \quad \bar{x} = \hat{x}.$$

**Доказательство.** Пусть

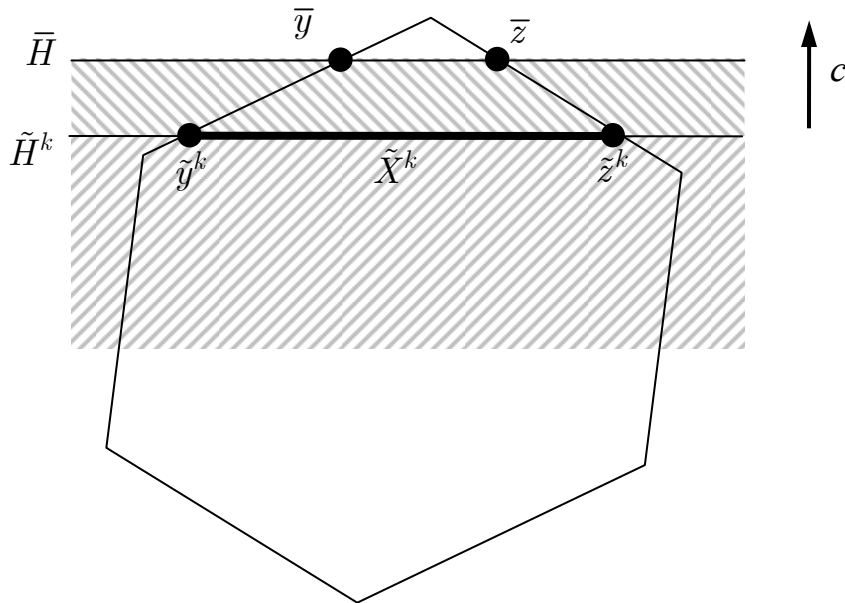
$$\lim_{k \rightarrow \infty} \text{dist}(\tilde{H}^k, \bar{H}) = 0. \quad (1.13)$$

В силу предположений, объявленных в разделе 1.2, имеем  $D \cap \bar{P} \setminus \bar{H} \neq \emptyset$ . Следовательно существуют два различных ребра  $Y$  и  $Z$  многогранника  $D$  такие, что

$$\bar{H} \cap Y \neq \emptyset, \bar{H} \cap Z \neq \emptyset, Y \cap \bar{H} \cap Z = \emptyset$$

(см. Рис. 2). В силу (1.13) и условия нормализации, откуда получаем

$$\exists k' : \forall k \geq k' : \tilde{H}^k \cap Y = \{\tilde{y}^k\}, \tilde{H}^k \cap Z = \{\tilde{z}^k\}, \tilde{y}^k \neq \tilde{z}^k. \quad (1.14)$$



**Рис. 2.** Задача ЛПА<sup>k</sup> имеет бесконечное множество решений.

Из (1.13) и условия нормализации следует, что последовательности  $\{\tilde{y}^k\}_{k=0}^{\infty}$  и  $\{\tilde{z}^k\}_{k=0}^{\infty}$  сходятся. Пусть

$$\{\tilde{y}^k\}_{k=0}^{\infty} \rightarrow \bar{y}, \quad \{\tilde{z}^k\}_{k=0}^{\infty} \rightarrow \bar{z}.$$

Очевидно, что

$$\{\bar{y}, \bar{z}\} \subset D \cap \bar{H} \quad (1.15)$$

и

$$\bar{y} \in Y, \bar{z} \in Z. \quad (1.16)$$

В силу непрерывности линейного функционала имеем

$$\lim_{k \rightarrow \infty} (c, \tilde{y}^k) = (c, \bar{y}), \quad \lim_{k \rightarrow \infty} (c, \tilde{z}^k) = (c, \bar{z}). \quad (1.17)$$

Так как задача ЛПФ имеет единственное решение, а задача ЛПА<sup>k</sup> для всех достаточно больших  $k$  имеет бесконечное множество решений  $\tilde{X}^k$ , то  $\exists k' : \forall k \geq k' : \tilde{X}^k = D \cap \tilde{H}^k$ . Сопоставляя это с (1.14) получаем

$$\exists k' : \forall k \geq k' : \{\tilde{y}^k, \tilde{z}^k\} \subset \tilde{X}^k. \quad (1.18)$$

Из (1.18) получаем

$$\exists k' : \forall k \geq k' : \tilde{\mu}^k = (c, \tilde{y}^k) = (c, \tilde{z}^k), \quad (1.19)$$

где  $\tilde{\mu}^k = \max\{(c, x) \mid x \in D, \tilde{\varphi}^k(x) \leq 0\}$ .

В соответствии с теоремой 1, из (1.13) и условия нормализации следует, что

$$\lim_{k \rightarrow \infty} \tilde{\mu}^k = \bar{\mu}, \quad (1.20)$$

где  $\bar{\mu} = (c, \bar{x})$ .

Из (1.17), (1.19) и (1.20) получаем

$$(c, \bar{y}) = \lim_{k \rightarrow \infty} (c, \tilde{y}^k) = \lim_{k \rightarrow \infty} \tilde{\mu}^k = \bar{\mu},$$

$$(c, \bar{z}) = \lim_{k \rightarrow \infty} (c, \tilde{z}^k) = \lim_{k \rightarrow \infty} \tilde{\mu}^k = \bar{\mu}.$$

Учитывая (1.15), отсюда заключаем, что  $\bar{y}, \bar{z}$  также являются решениями задачи ЛПНО. Из предположения о единственности решения задачи ЛПНО следует, что  $\bar{y} = \bar{x} = \bar{z}$ . Вместе с (1.16) это означает, что

$$D \cap \bar{H} = \{\bar{x}\}. \quad (1.21)$$

Так как  $D \cap \bar{P} \setminus \bar{H} \neq \emptyset$ , то из (1.21) следует, что  $D \subset \bar{P}$ , откуда в свою очередь следует, что  $\bar{x}$  является также решением задачи ЛПФ, то есть  $\bar{x} = \hat{x}$ . Лемма доказана.

**Лемма 2.** В контексте алгоритма  $\mathfrak{L}$  справедливо следующее утверждение:

$$\lim_{k \rightarrow \infty} \text{dist}(\tilde{H}^k, \bar{H}) = 0 \quad \Rightarrow \quad \{\tilde{x}^k\}_{k=0}^{\infty} \rightarrow \bar{x}.$$

**Доказательство.** Пусть

$$\lim_{k \rightarrow \infty} \text{dist}(\tilde{H}^k, \bar{H}) = 0. \quad (1.22)$$

Предположим, что  $\bar{x} \notin \bar{H}$ . Это означает, что  $\bar{x}$  является решением задачи ЛПФ, то есть  $\bar{x} = \hat{x}$  (см. Рис. 3). Получили противоречие с условием (1.6). Следовательно  $\bar{x} \in \bar{H}$  (см. Рис. 4). Имеются три взаимоисключающих случая:

1.  $\exists k' : \forall k \geq k' : \tilde{x}^k \notin \tilde{H}^k$ ,
2.  $\exists k' : \forall k \geq k' : \tilde{x}^k \in \tilde{H}^k$ ,
3.  $\forall k : \exists k_1, k_2 \geq k : \tilde{x}^{k_1} \in \tilde{H}^{k_1}, \tilde{x}^{k_2} \notin \tilde{H}^{k_2}$ .

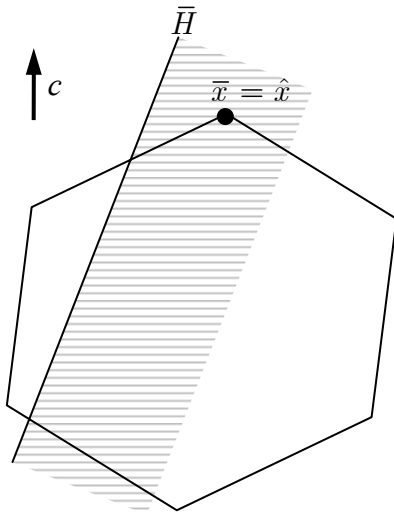


Рис. 3. Случай  $\bar{x} \notin \bar{H}$  ( $\bar{x} = \hat{x}$ ).

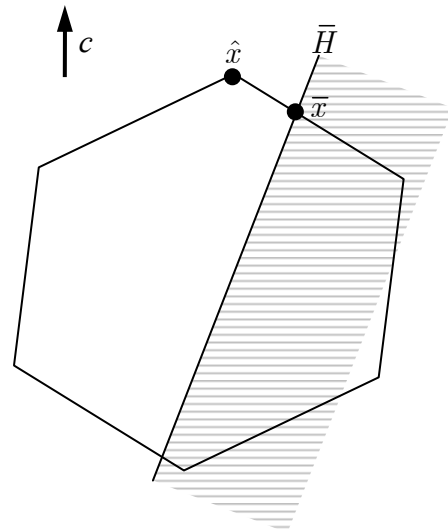


Рис. 4. Случай  $\bar{x} \in \bar{H}$  ( $\bar{x} \neq \hat{x}$ ).

Рассмотрим каждый случай.

Случай 1. Пусть

$$\exists k' : \forall k \geq k' : \tilde{x}^k \notin \tilde{H}^k. \quad (1.23)$$

Покажем, что в условиях нашей леммы соотношение (1.23) не может иметь место. В силу единственности решения задачи ЛПФ из (1.23) следует

$$\exists k' : \forall k \geq k' : \tilde{x}^k = \hat{x}. \quad (1.24)$$

В совокупности (1.23) и (1.24) дают

$$\exists k' : \forall k \geq k' : \tilde{\varphi}^k(\hat{x}) < 0. \quad (1.25)$$

Предположим сначала, что  $\bar{\varphi}(\hat{x}) \leq 0$ . Тогда  $\bar{x} = \hat{x}$ , что противоречит условию (1.6). Предположим теперь, что

$$\bar{\varphi}(\hat{x}) > 0. \quad (1.26)$$

В соответствии с определением алгоритма  $\mathfrak{L}$  существует  $x^0 \in \mathbb{B}^0$  :

$$\forall k : \tilde{\varphi}^k(x^0) \leq 0 \quad (1.27)$$

и

$$\bar{\varphi}(x^0) < 0. \quad (1.28)$$

Из (1.26) и (1.28) следует, что точки  $\hat{x}$  и  $x^0$  лежат по разные стороны от гиперплоскости  $\bar{H}$ . Поскольку  $\lim_{k \rightarrow \infty} \text{dist}(\tilde{H}^k, \bar{H}) = 0$ , то для всех достаточно больших  $k$  точки  $\hat{x}$  и  $x^0$  будут также лежать по разные стороны от гиперплоскости  $\tilde{H}^k$ . Это означает, что для всех достаточно больших  $k$  функция  $\tilde{\varphi}^k(x)$  будет иметь разные знаки в точках  $\hat{x}$  и  $x^0$ . Учитывая это, из (1.25) получаем  $\exists k' : \forall k \geq k' : \tilde{\varphi}^k(x^0) > 0$ , что противоречит (1.27). Это означает, что условие (1.26) также не может иметь место. Таким образом, в условиях нашей леммы предположение (1.23) всегда ложно.

*Случай 2.* Допустим, что

$$\exists k' : \forall k \geq k' : \tilde{x}^k \in \tilde{H}^k \quad (1.29)$$

Пусть  $\tilde{X}^k = \{x \mid x \in D, \tilde{\varphi}^k(x) \leq 0, (c, x) = \tilde{\mu}^k\}$  обозначает множество всех решений задачи ЛПА<sup>k</sup>:

$$\tilde{\mu}^k = \max\{(c, x) \mid x \in D, \tilde{\varphi}^k(x) \leq 0\}. \quad (1.30)$$

Имеются три взаимоисключающие возможности:

$$\exists k' : \forall k \geq k' : \tilde{X}^k \setminus \{\tilde{x}^k\} = \emptyset,$$

$$\exists k' : \forall k \geq k' : \tilde{X}^k \setminus \{\tilde{x}^k\} \neq \emptyset,$$

$$\forall k : \exists k_1, k_2 \geq k : \tilde{X}^{k_1} \setminus \{\tilde{x}^{k_1}\} \neq \emptyset, \tilde{X}^{k_2} \setminus \{\tilde{x}^{k_2}\} = \emptyset.$$

Рассмотрим каждую из возможностей.

*Возможность 1.* Предположим, что

$$\exists k' : \forall k \geq k' : \tilde{X}^k \setminus \{\tilde{x}^k\} = \emptyset.$$

Покажем, что в этом случае  $\bar{x}$  и  $\tilde{x}^k$  будут принадлежать некоторому общему ребру  $Y$  многогранника  $D$  для всех достаточно больших  $k$ . Предположим противное:

$$\forall k' : \exists k \geq k' : \bar{x} \in Y, \tilde{x}^k \in Z, \bar{x} \notin Z, \tilde{x}^k \notin Y, \quad (1.31)$$

где  $Y, Z$  – ребра многогранника  $D$  (см. Рис. 5). Пусть  $\bar{\mu} = (c, \bar{x})$ . Положим

$$\varepsilon = \min\{|\bar{\mu} - \hat{\mu}_i| \mid \hat{\mu}_i = (c, \hat{x}_i), \hat{x}_i - \text{вершина } D, \hat{x}_i \neq \bar{x}\} > 0. \quad (1.32)$$

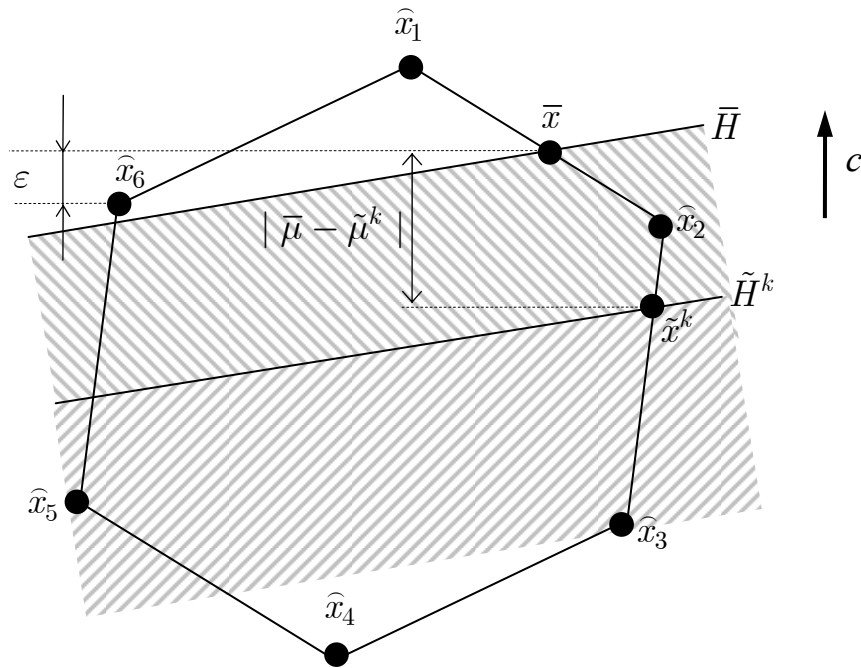


Рис. 5. Ситуация, когда  $\bar{x}$  и  $\tilde{x}^k$  принадлежат разным ребрам.

Из (1.31) и (1.32) получаем  $\forall k' : \exists k \geq k' : |\bar{\mu} - \tilde{\mu}^k| \geq \varepsilon$ . С другой стороны, из (1.22) и теоремы 1 следует, что  $\exists k' : \forall k \geq k' : |\bar{\mu} - \tilde{\mu}^k| < \varepsilon$ . Получили противоречие. Следовательно

$$\exists k' : \forall k \geq k' : \bar{x}, \tilde{x}^k \in Y, \quad (1.33)$$

где  $Y$  – ребро многогранника  $D$ . Вместе с (1.22) и (1.29) это дает  $\lim_{k \rightarrow \infty} \|\bar{x} - \tilde{x}^k\| = 0$ , то есть  $\{\tilde{x}^k\}_{k=0}^{\infty} \rightarrow \bar{x}$ , что и требовалось получить.

*Возможность 2.* Предположим, что

$$\exists k' : \forall k \geq k' : \tilde{X}^k \setminus \{\tilde{x}^k\} \neq \emptyset.$$

Это означает, что задача ЛПА<sup>k</sup> (1.30) имеет бесконечно много решений для всех достаточно больших  $k$ . Тогда в соответствии с леммой 1 получаем  $\bar{x} = \hat{x}$ , что противоречит условию (1.6).

*Возможность 3.* Предположим, что

$$\forall k : \exists k_1, k_2 \geq k : \tilde{X}^{k_1} \setminus \{\tilde{x}^{k_1}\} \neq \emptyset, \tilde{X}^{k_2} \setminus \{\tilde{x}^{k_2}\} = \emptyset. \quad (1.34)$$

Тогда из последовательности  $\{\tilde{x}^k\}_{k=0}^{\infty}$  можно выделить бесконечную подпоследовательность  $\{\tilde{x}^{k'}\}_{k'=0}^{\infty}$ , такую, что

$$\forall k' : \tilde{X}^{k'} \setminus \{\tilde{x}^{k'}\} \neq \emptyset. \quad (1.35)$$

Учитывая, что из (1.22) следует  $\lim_{k' \rightarrow \infty} \text{dist}(\tilde{H}^{k'}, \bar{H}) = 0$ , мы можем по аналогии с возможностью 2 показать, что условие (1.35), а следовательно и условие (1.34) не может иметь место. Таким образом, для случая 2 лемма также доказана.

*Случай 3.* Пусть  $\forall k : \exists k_1, k_2 \geq k : \tilde{x}^{k_1} \in \tilde{H}^{k_1}, \tilde{x}^{k_2} \notin \tilde{H}^{k_2}$ . Разделим последовательность  $\{\tilde{x}^k\}_{k=0}^\infty$  на две бесконечные подпоследовательности  $\{\tilde{y}^k\}_{k=0}^\infty$  и  $\{\tilde{z}^k\}_{k=0}^\infty$  такие, что  $\forall k : \tilde{y}^k \in \tilde{H}^k$  и  $\forall k : \tilde{z}^k \notin \tilde{H}^k$ . Тогда, рассуждая по аналогии со случаями 1 и 2, получим  $\{\tilde{y}^k\}_{k=0}^\infty \rightarrow \bar{x}$  и  $\{\tilde{z}^k\}_{k=0}^\infty \rightarrow \bar{x}$ . Откуда следует  $\{\tilde{x}^k\}_{k=0}^\infty \rightarrow \bar{x}$ . Лемма полностью доказана.

**Лемма 3.** Применительно к алгоритму  $\mathfrak{L}$  хотя бы одно из следующих условий истинно:

$$\{\tilde{x}^k\}_{k=0}^\infty \rightarrow \bar{x}; \quad (1.36)$$

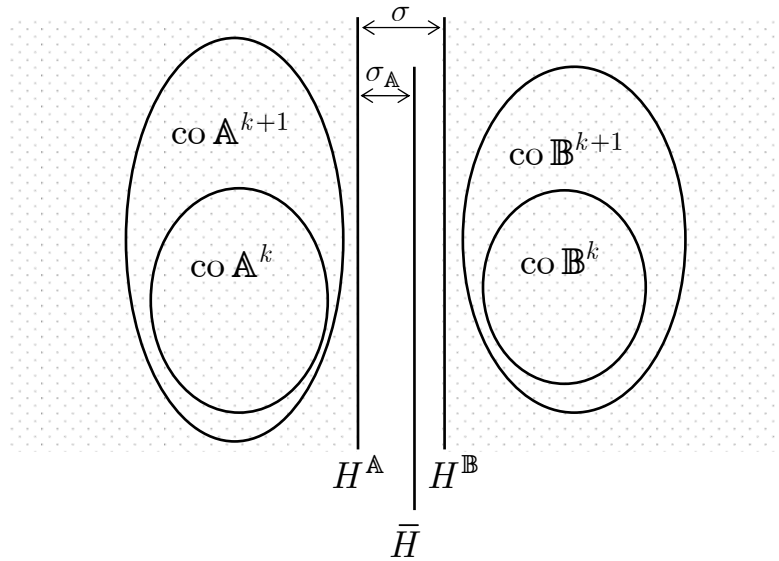
$$\lim_{k \rightarrow \infty} \text{dist}(\text{co } \mathbb{A}^k, \text{co } \mathbb{B}^k) = 0 \quad (1.37)$$

(здесь  $\text{co}$  – выпуклая оболочка).

*Доказательство.* Предположим, что условие (1.36) не выполняется. Покажем, что в этом случае условие (1.37) истинно. Пусть  $\sigma_k = \text{dist}(\text{co } \mathbb{A}^k, \text{co } \mathbb{B}^k)$ . Так как  $\mathbb{A}^{k+1} \supset \mathbb{A}^k, \mathbb{B}^{k+1} \supset \mathbb{B}^k$ , то  $\forall k : \sigma_{k+1} \leq \sigma_k$ . Следовательно существует  $\lim_{k \rightarrow \infty} \sigma_k = \sigma \geq 0$ . Покажем, что соотношение  $\sigma > 0$  невозможно. Для этого предположим, от противного, что  $\sigma > 0$ . В этом случае существуют гиперплоскости  $H^{\mathbb{A}} = \{x \mid \varphi_{\mathbb{A}}(x) = 0\}$  и  $H^{\mathbb{B}} = \{x \mid \varphi_{\mathbb{B}}(x) = 0\}$ , удовлетворяющие условию нормализации, такие, что:

$$\text{dist}(H^{\mathbb{A}}, H^{\mathbb{B}}) = \sigma; \quad \text{dist}(\bar{H}, H^{\mathbb{A}}) \leq \sigma; \quad \text{dist}(\bar{H}, H^{\mathbb{B}}) \leq \sigma; \quad (1.38)$$

$$\forall x \in \mathbb{A}^k : \varphi_{\mathbb{A}}(x) \geq 0; \quad \forall x \in \mathbb{B}^k : \varphi_{\mathbb{B}}(x) \leq 0 \quad (\forall k); \quad (1.39)$$



**Рис. 6.** Сближение выпуклых оболочек.

$$\lim_{k \rightarrow \infty} \text{dist}(\text{co } \mathbb{A}^k, H^{\mathbb{A}}) = 0; \quad (1.40)$$

$$\lim_{k \rightarrow \infty} \text{dist}(\text{co } \mathbb{B}^k, H^{\mathbb{B}}) = 0 \quad (1.41)$$

(см. Рис. 6).

Так как мы предположили, что  $\sigma > 0$ , то справедливо хотя бы одно из следующих условий:

- 1)  $\text{dist}(H^{\mathbb{A}}, \bar{H}) > 0$ ,
- 2)  $\text{dist}(H^{\mathbb{B}}, \bar{H}) > 0$ .

Докажем лемму для каждого случая.

*Случай (I).* Пусть  $\text{dist}(H^{\mathbb{A}}, \bar{H}) = \sigma_{\mathbb{A}} > 0$ . Тогда в силу (1.40) имеем

$$\lim_{k \rightarrow \infty} \text{dist}(\text{co } \mathbb{A}^k, \bar{H}) = \sigma_{\mathbb{A}} > 0.$$

Имеются две взаимоисключающие возможности:

1.  $\exists k' : \forall k \geq k' : (c, \tilde{x}^k) \notin \{(c, x) \mid x \in \mathbb{A}^k \cup \mathbb{B}^k \cup \bar{H}\}$ ,
2.  $\forall k' : \exists k \geq k' : (c, \tilde{x}^k) \in \{(c, x) \mid x \in \mathbb{A}^k \cup \mathbb{B}^k \cup \bar{H}\}$ .

Рассмотрим каждую из возможностей.

*Возможность 1.* Предположим, что

$$\exists k' : \forall k \geq k' : (c, \tilde{x}^k) \notin \{(c, x) \mid x \in \mathbb{A}^k \cup \mathbb{B}^k \cup \bar{H}\}. \quad (1.42)$$

По построению алгоритма  $\mathfrak{L}$  отсюда следует

$$\exists k' : \forall k \geq k' : \tilde{x}^k \in \mathbb{A}^{k+1} \cup \mathbb{B}^{k+1} \cup \bar{H}. \quad (1.43)$$

Имеются два взаимоисключающих случая:

- a)  $\exists k' : \forall k \geq k' : \tilde{x}^k \in \tilde{H}^k$ ,
- b)  $\forall k' : \exists k \geq k' : \tilde{x}^k \notin \tilde{H}^k$ .

Рассмотрим оба случая.

a) Пусть

$$\exists k' : \forall k \geq k' : \tilde{x}^k \in \tilde{H}^k. \quad (1.44)$$

Тогда

$$\exists k' : \forall k > k' : \text{dist}(\tilde{H}^{k+1}, \tilde{H}^k) > 0. \quad (1.45)$$

Без ограничения общности мы можем считать, что все  $\tilde{x}^k$  лежат по одну сторону от  $\bar{H}$ . Тогда по построению алгоритма  $\mathfrak{L}$  с учетом (1.43) и (1.44) имеем

$$\text{dist}(\tilde{H}^{k+1}, \bar{H}) \leq \text{dist}(\tilde{H}^k, \bar{H}).$$

Учитывая (1.45), отсюда получаем

$$\text{dist}(\tilde{H}^{k+1}, \bar{H}) < \text{dist}(\tilde{H}^k, \bar{H}).$$

Следовательно,

$$\lim_{k \rightarrow \infty} \text{dist}(\tilde{H}^k, \bar{H}) = 0.$$

В соответствии с леммой 2 это влечет

$$\{\tilde{x}^k\}_{k=0}^{\infty} \rightarrow \bar{x},$$

что противоречит нашему предположению, что условие (1.36) не выполняется.

b) Пусть

$$\forall k' : \exists k \geq k' : \tilde{x}^k \notin \tilde{H}^k. \quad (1.46)$$

Если  $\tilde{x}^k \notin \tilde{H}^k$ , то  $\tilde{x}^k$  является одной из вершин многогранника  $D$ . Так как количество вершин конечно, то из (1.46) следует

$$\forall k' : \exists k \geq k' : \tilde{x}^k \in \mathbb{A}^k \cup \mathbb{B}^k \quad (1.47)$$

Однако (1.47) противоречит предположению (1.42).

*Возможность 2.* Предположим, что

$$\forall k : \exists k' \geq k : (c, \tilde{x}^k) \in \{(c, x) \mid x \in \mathbb{A}^k \cup \mathbb{B}^k \cup \bar{H}\}. \quad (1.48)$$

В силу (1.40), для любого наперед заданного положительного  $\delta$ , при достаточно большом  $k$ ,  $\exists x' \in \mathbb{A}^k$ :

$$\text{dist}(\{x'\}, H^{\mathbb{A}}) < \delta.$$

Положим  $\delta = \frac{\varepsilon}{2}$ , где

$$\varepsilon = \frac{\min(\text{dist}(H^{\mathbb{A}}, \bar{H}), \text{dist}(H^{\mathbb{B}}, \bar{H}))}{2}. \quad (1.49)$$

В соответствии с предположением (1.48) алгоритм  $\mathfrak{L}$  будет выполнять процедуру рандомизации бесконечное число раз. При этом в соответствии с (1.11) в качестве радиуса рандомизации мы можем взять положительную константу  $\varepsilon$ , задаваемую формулой (1.49). Но тогда

$$\sigma_{\mathbb{A}} = \lim_{k \rightarrow \infty} \text{dist}(\text{co } \mathbb{A}^k, \bar{H}) \leq \text{dist}(S^{x'}, \bar{H}) < \text{dist}(H^{\mathbb{A}}, \bar{H}) - \frac{\varepsilon}{2} = \sigma_{\mathbb{A}} - \frac{\varepsilon}{2},$$

где  $S^{x'}$  – сфера с центром в точке  $x'$  и радиусом  $\varepsilon$ . Получили противоречие.

*Случай (II).* Доказывается аналогично случаю (I). Лемма доказана.

**Доказательство теоремы 2.** В соответствии с леммой 3 нам необходимо рассмотреть случай, когда

$$\lim_{k \rightarrow \infty} \text{dist}(\text{co } \mathbb{A}^k, \text{co } \mathbb{B}^k) = 0. \quad (1.50)$$

Так как гиперплоскости  $\tilde{H}^k$  и  $\bar{H}$  являются разделяющими для множеств  $\mathbb{A}^k$  и  $\mathbb{B}^k$ , то, в силу условия нормализации, из (1.50) непосредственно следует, что

$$\lim_{k \rightarrow \infty} \text{dist}(\tilde{H}^k, \bar{H}) = 0.$$

Отсюда по лемме 2 получаем  $\{\tilde{x}^k\}_{k=0}^{\infty} \rightarrow \bar{x}$ . Теорема доказана.

### 1.6. Реализационные аспекты алгоритма $\mathfrak{L}$

Реализация алгоритма  $\mathfrak{L}$  непосредственно в том виде, как он был описан в разделе 1.4, на практике не всегда возможна. Прежде всего это относится к условию нормализации. Во многих случаях эксперт не в состоянии определить нормаль к гиперплоскости, задаваемой неформализованным ограничением. Следующая теорема показывает, что при определенных обстоятельствах условием нормализации можно пренебречь.

**Теорема 3.** Пусть  $L^\varepsilon$  – плоский слой толщины  $\varepsilon > 0$ , серединой которого является гиперплоскость  $\bar{H}$ , задаваемая неформализованным ограничением. Обозначим через  $\mathfrak{L}'$  алгоритм, получающийся из алгоритма  $\mathfrak{L}$  путем снятия ограничения на  $\tilde{\varphi}^k$ , связанного с выполнением условия нормализации. Тогда, если для алгоритма  $\mathfrak{L}'$  выполняется условие

$$\forall k' : \exists k \geq k' : \tilde{x}^k \in \mathbb{A}^k \cup \mathbb{B}^k \cup \bar{H}, \quad (1.51)$$

то

$$\forall \varepsilon > 0 : \exists k' : \forall k \geq k' : (\tilde{H}^k \cap D) \subset (L^\varepsilon \cap D).$$

**Доказательство.** Пусть  $C$  –  $n$ -мерный цилиндр с осью  $\bar{\eta}$ , обладающий свойством  $D \subset C$ . Пусть  $S$  – проекция  $C$  на гиперплоскость  $\bar{H}$ . Обозначим через  $d$  диаметр  $(n-1)$ -мерной сферы  $S$ . Выберем  $(n-1)$  пар точек  $\{x^i, y^i\} \subset S$ , удовлетворяющих следующим требованиям:

$$\begin{aligned} \|y^i - x^i\| &= d; \\ (y^i - x^i, y^j - x^j) &= 0, \quad i \neq j. \end{aligned}$$

Построим  $n$ -мерные сферы  $s_x^i$  и  $s_y^i$  с центрами в точках  $x^i$  и  $y^i$  соответ-

ственно ( $i = 1, \dots, n-1$ ) и радиусом  $\frac{d\varepsilon/2}{\sqrt{d^2 + (\varepsilon/2)^2}}$ .

Условие (1.51) означает, что алгоритм  $\mathfrak{L}'$  бесконечное число раз будет выполнять процедуру рандомизации. Следовательно,

$$\begin{aligned} \exists k' : \mathbb{A}^{k'} \cap s_x^i \neq \emptyset, \mathbb{A}^{k'} \cap s_y^i \neq \emptyset, \\ \mathbb{B}^{k'} \cap s_x^i \neq \emptyset, \mathbb{B}^{k'} \cap s_y^i \neq \emptyset \quad (i = 1, \dots, n-1). \end{aligned}$$

Это означает, что

$$\forall k > k' : \tilde{H}^k \cap s_x^i \neq \emptyset, \tilde{H}^k \cap s_y^i \neq \emptyset \quad (i = 1, \dots, n-1).$$

В соответствии со способом построения  $s_x^i$  и  $s_y^i$  отсюда получаем

$$\forall k > k' : (\tilde{H}^k \cap C) \subset (L^\varepsilon \cap C).$$

Учитывая, что  $D \subset C$ , отсюда следует

$$\forall k > k' : (\tilde{H}^k \cap D) \subset (L^\varepsilon \cap D).$$

Теорема доказана.

Теорема 3 показывает, что, если при поиске приближенного решения с помощью алгоритма  $\mathfrak{L}'$  регулярно выполняется рандомизация, то при  $k$ , стремящемся к бесконечности,  $\tilde{H}^k$  «стремится стать параллельной» к  $\bar{H}$ . Нами была проведена серия вычислительных экспериментов по решению различных задач ЛПНО с помощью метода, базирующегося на алгоритме  $\mathfrak{L}'$  (см. гл. 4). Во всех экспериментах на завершающей стадии вычислений наблюдалось регулярное обращение к процедуре рандомизации. В соответствии с этим на практике мы можем пренебречь условием нормализации без потери эффективной сходимости.

Отметим, что при реализации алгоритма  $\mathfrak{L}$  в виде компьютерной программы необходимо вместо условия (1.10) использовать условие  $(c, \tilde{x}^k) \notin \{(c, x) \mid x \in \mathbb{A}^k \cup \mathbb{B}^k\}$ , не требующее знания неформализованного ограничения. Учитывая приближенный характер компьютерных вычислений, мы вправе считать эти два условия эквивалентными.

В заключение этого раздела рассмотрим вопрос реализации условия (1.11) при написании компьютерной программы. В качестве радиуса рандомизации следует взять некоторую константу  $\varepsilon > 0$ . При этом для сохранения корректности вычислительного процесса необходимо все образцы, полученные в результате выполнения процедуры рандомизации, отпра-

лять на экспертизу. Очевидно, что в этом случае мы сохраним эффективную сходимость. Для уменьшения количества экспертиз процедуру рандомизации следует применять только к образцу  $\tilde{x}^k$ , полученному на последнем шаге.

### 1.7. Случай нескольких неформализованных ограничений

Пусть модель экономической ситуации описывается в виде задачи линейной оптимизации, в которой нам не удалось полностью формализовать несколько ограничений. Такую модель мы можем представить в виде следующей задачи линейного программирования с несколькими неформализованными ограничениями (ЛПНО\*) в пространстве  $\mathbb{R}^n$ :

$$\bar{x} \in \text{Arg max} \{ (c, x) \mid Ax \leq b, \bar{\varphi}_1(x) \leq 0, \dots, \bar{\varphi}_q(x) \leq 0 \}. \quad (1.52)$$

Здесь неравенства  $\bar{\varphi}_1(x) \leq 0, \dots, \bar{\varphi}_q(x) \leq 0$  играют роль неформализованных ограничений в том смысле, что нам не известен вид функций  $\bar{\varphi}_i$  ( $i = 1, \dots, q$ ). Будем предполагать, что функции  $\bar{\varphi}_i$  принадлежат к классу аффинных функций, и что задача (1.52) имеет единственное решение. Пусть  $\bar{P}_i = \{x \mid \bar{\varphi}_i(x) \leq 0\}$  – множество значений, удовлетворяющих неформализованному ограничению  $\bar{\varphi}_i(x) \leq 0$ . Так как  $\bar{\varphi}_i$  является аффинной функцией, то  $\bar{P}_i$  представляет собой замкнутое полупространство. Пусть  $\bar{P}^* = \bigcap_{i=1}^q \bar{P}_i$ . Тогда допустимое множество задачи ЛПНО\* будет иметь вид  $D \cap \bar{P}^*$ . Будем предполагать, что  $D \cap \bar{P}^*$  содержит внутренние точки.

Имеет место следующее обобщение теоремы 1 (стр. 15) для случая нескольких неформализованных ограничений.

**Теорема 4.** Задача ЛПНО\* (1.52) является устойчивой по  $\bar{\varphi}_1, \dots, \bar{\varphi}_q$ .

Доказательство аналогично доказательству теоремы 1.

Покажем теперь, как решение задачи ЛПНО\* может быть сведено к решению нескольких задач ЛПНО с одним неформализованным ограничением.

Рассмотрим следующую последовательность *независимых* задач ЛПНО, каждая из которых имеет только одно неформализованное ограничение

$$\text{ЛПНО}_1 : \quad \bar{x}_1 = \arg \max \{(c, x) \mid Ax \leq b, \bar{\varphi}_1(x) \leq 0\};$$

...

$$\text{ЛПНО}_q : \quad \bar{x}_q = \arg \max \{(c, x) \mid Ax \leq b, \bar{\varphi}_q(x) \leq 0\}$$

(предполагается, что каждая из этих задач имеет единственное решение).

Используя алгоритм  $\mathfrak{L}$  из раздела 1.4, мы можем построить для каждой задачи ЛПНО<sub>*i*</sub> ( $i = 1, \dots, q$ ) последовательность аффинных *разделяющих* функций  $\tilde{\varphi}_i^1, \dots, \tilde{\varphi}_i^k$ . Сконструируем следующую задачу линейного программирования:

$$\text{ЛП}^k : \quad \tilde{x}^k \in \text{Arg max} \{(c, x) \mid Ax \leq b, \tilde{\varphi}_1^k(x) \leq 0, \dots, \tilde{\varphi}_q^k(x) \leq 0\}. \quad (1.53)$$

**Теорема 5.** Пусть задача ЛПНО\* (1.52) имеет единственное решение  $\bar{x}$ . Пусть каждая задача ЛПНО<sub>*i*</sub> ( $1 \leq i \leq q$ ) также имеет единственное решение  $\bar{x}_i$ , причем  $\bar{x}_i \neq \hat{x}$ , где  $\hat{x} = \arg \max \{(c, x) \mid Ax \leq b\}$  – единственное решение задачи ЛПФ. Пусть  $\tilde{x}^k$  – решение задачи ЛП<sup>*k*</sup>. Тогда

$$\{\tilde{x}^k\}_{k=0}^{\infty} \rightarrow \bar{x},$$

то есть при достаточно большом  $k$  в качестве приближенного решения задачи ЛПНО\* (1.52) можно взять решение задачи ЛП<sup>*k*</sup> (1.53).

**Доказательство.** Пусть  $\tilde{\varphi}_i^1, \dots, \tilde{\varphi}_i^k$  – последовательность разделяющих функций, получаемых в ходе выполнения алгоритма  $\mathfrak{L}$  применительно к задаче ЛПНО<sub>*i*</sub> ( $1 \leq i \leq q$ ). Пусть  $\tilde{x}_i^0, \tilde{x}_i^1, \dots, \tilde{x}_i^k$  – порождаемая ими

последовательность приближенных решений задачи ЛПНО<sub>*i*</sub>. В соответствии с теоремой 2 (стр. 20) имеем

$$\{\tilde{x}_i^k\}_{k=0}^\infty \rightarrow \bar{x}_i. \quad (1.54)$$

Так как  $\bar{x}_i \neq \hat{x}$  и  $\bar{x}_i$  – единственное решение задачи ЛПНО<sub>*i*</sub>, то  $\bar{x}_i \in \bar{H}_i$ , где  $\bar{H}_i = \{x \mid \bar{\varphi}_i(x) = 0\}$ . С учетом этого в силу условия нормализации (стр. 17) из (1.54) следует, что последовательность функций  $\tilde{\varphi}_i^1, \dots, \tilde{\varphi}_i^k$  сходится к  $\bar{\varphi}_i$  в том смысле, что

$$\lim_{k \rightarrow \infty} \text{dist}(\tilde{H}_i^k, \bar{H}_i) = 0, \quad (1.55)$$

где  $\tilde{H}_i^k = \{x \mid \tilde{\varphi}_i^k(x) = 0\}$ .

Поскольку  $\bar{\varphi}_i$  принадлежит к классу аффинных функций, мы можем представить уравнение  $\bar{\varphi}_i(x) = 0$  в виде

$$(\bar{d}_i, x) = \alpha, \quad \alpha \in \{0; 1\}. \quad (1.56)$$

Тогда, в силу (1.55) существует  $k'_i$  такой, что для всех  $k > k'_i$  уравнение  $\tilde{\varphi}_i^k(x) = 0$  можно представить в виде  $(\tilde{d}_i^k, x) = \alpha$ , причем  $\|\bar{d}_i - \tilde{d}_i^k\| < \varepsilon/\sqrt{q}$  для любого наперед заданного числа  $\varepsilon > 0$ . Положим  $k' = \max\{k'_1, \dots, k'_q\}$ .

Тогда для всех  $k > k'$  имеем

$$\|[\bar{d}_1, \dots, \bar{d}_q] - [\tilde{d}_1^k, \dots, \tilde{d}_q^k]\| = \sqrt{\|\bar{d}_1 - \tilde{d}_1^k\|^2 + \dots + \|\bar{d}_q - \tilde{d}_q^k\|^2} < \sqrt{q(\varepsilon/\sqrt{q})^2} = \varepsilon.$$

Отсюда по теореме 4 получаем  $\lim_{k \rightarrow \infty} |(c, \bar{x}) - (c, \tilde{x}^k)| = 0$ . Так как  $\bar{x}$  – единственное решение задачи ЛПНО\*, это равносильно

$$\{\tilde{x}^k\}_{k=0}^\infty \rightarrow \bar{x},$$

что и требовалось доказать.

В формулировке теоремы 5 мы исходили из предположения, что *все* задачи ЛПНО<sub>*i*</sub> имеют решение  $\bar{x}_i$ , не совпадающее с решением  $\hat{x}$  задачи

ЛПФ. На практике это требование мы можем выполнить следующим образом.

На первом этапе находим приближенные значения коэффициентов и свободных членов для всех неформализованных ограничений  $\bar{\varphi}_i(x) \leq 0$ , удовлетворяющих условию  $\bar{x}_i \neq \hat{x}$  (каждое ограничение вычисляется *независимо* от других). Заметим, что если ни одно неформализованное ограничение не удовлетворяет условию  $\bar{x}_i \neq \hat{x}$ , то решением задачи ЛПНО\* будет  $\hat{x}$ .

На втором этапе добавляем к системе ограничений задачи ЛПФ найденные на первом этапе *приближения* неформализованных ограничений и полагаем  $\hat{x}$  равным решению расширенной задачи ЛПФ. Из оставшихся неформализованных ограничений снова выбираем все, удовлетворяющие условию  $\bar{x}_i \neq \hat{x}$ , и находим их приближенный вид.

Процесс заканчивается, когда либо все неформализованные ограничения будут найдены, либо останутся несколько ограничений, для которых  $\bar{x}_i = \hat{x}$  (эти ограничения можно отбросить). Последняя расширенная задача ЛПФ даст приближенное решение исходной задачи ЛПНО\*.

## ГЛАВА 2. АЛГОРИТМ ЛП-ДА

### 2.1. Общая схема алгоритма ЛП-ДА

Для решения задачи линейного программирования с неформализованным ограничением может быть использован следующий общий алгоритм ЛП-ДА, основанный на синтезе алгоритмов линейного программирования и дискриминантного анализа (см. Рис. 7).

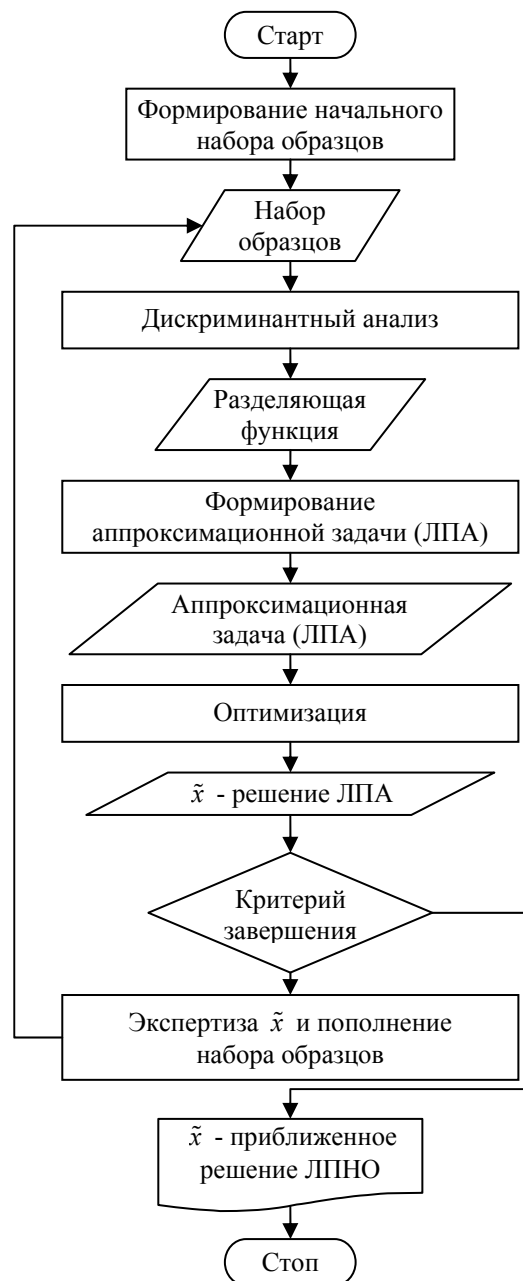


Рис. 7. Алгоритм решения ЛП-ДА.

На первом шаге формируется начальный набор образцов  $\mathfrak{P} = \{\mathbb{A}, \mathbb{B}\}$  путем задания пары конечных непересекающихся множеств  $\mathbb{A}$  и  $\mathbb{B}$  из пространства  $\mathbb{R}^n$ , удовлетворяющих условиям

$$\begin{cases} \epsilon(x) = 1, & \forall x \in \mathbb{A}; \\ \epsilon(x) = -1, & \forall x \in \mathbb{B}. \end{cases}$$

Существуют различные способы выбора точек для начального набора образцов. Это могут быть точки, полученные в результате предшествующего опыта, либо набор точек, заданных экспертом, либо множество точек, генерируемых некоторым специальным алгоритмом. Пример такого алгоритма будет приведен ниже.

На следующем шаге с помощью какого-либо метода дискриминантного анализа [17] строится аффинная функция  $\tilde{\varphi}$ , разделяющая множества  $\mathbb{A}$  и  $\mathbb{B}$ :

$$\begin{cases} \tilde{\varphi}(x) > 0, & \forall x \in \mathbb{A}; \\ \tilde{\varphi}(x) < 0, & \forall x \in \mathbb{B}. \end{cases}$$

Такая функция существует, так как мы предположили, что неформализованное ограничение  $\bar{\varphi}(x) \leq 0$  определяется аффинной функцией  $\bar{\varphi}$ . В качестве метода дискриминантного анализа могут фигурировать, например, метод линейной коррекции [17, 33], метод комитетов [26] или фейеровские методы [12, 15].

Далее с использованием полученной разделяющей функции строится аппроксимационная задача линейного программирования (ЛПА):

$$\tilde{x} \in \text{Arg max } \{(c, x) \mid Ax \leq b, \tilde{\varphi}(x) \leq 0\}. \quad (2.1)$$

Аппроксимационная задача решается в блоке *оптимизации* одним из методов линейного программирования. Здесь может использоваться, например, симплекс-метод [7, 52, 62] или методы фейеровского типа [12, 15].

С помощью некоторого критерия завершения итерационного процесса проверяется, является ли полученное решение  $\tilde{x}$  допустимым прибли-

жением к точному решению  $\bar{x}$ . Ниже мы приведем конкретный пример критерия завершения.

Если точка  $\tilde{x}$  не удовлетворяет критерию завершения, то она направляется в блок *экспертизы*, где к ней применяется функция эксперта  $\epsilon$ . Если  $\epsilon(\tilde{x}) = 1$ , то точка  $\tilde{x}$  добавляется в множество  $\mathbb{A}$ , если  $\epsilon(\tilde{x}) = -1$ , то точка  $\tilde{x}$  добавляется в множество  $\mathbb{B}$ . После этого описанный процесс повторяется применительно к расширенному набору образцов.

При проведении экспертизы и пополнении набора образцов могут возникать коллизии следующих трех типов:

1.  $\epsilon(\tilde{x}) = 0$ ;
2.  $\epsilon(\tilde{x}) = 1$  и  $\tilde{x} \in \mathbb{A}$  (добавляемый образец уже присутствует в множестве  $\mathbb{A}$ );
3.  $\epsilon(\tilde{x}) = -1$  и  $\tilde{x} \in \mathbb{B}$  (добавляемый образец уже присутствует в множестве  $\mathbb{B}$ ).

В этом случае к  $\tilde{x}$  применяется *процедура рандомизации* [69]. Точки, полученные в результате процедуры рандомизации, вновь подвергаются экспертизе и используются для пополнения набора образцов.

Реализация описанного алгоритма в виде программы требует решения следующих задач:

- 1) формирование начального набора образцов;
- 2) выбор критерия завершения итерационного процесса;
- 3) определение процедуры рандомизации;
- 4) обеспечение быстрой сходимости к точному решению;
- 5) решение проблемы погрешности вычислений.

В этой главе мы рассмотрим возможные подходы к решению указанных задач.

## 2.2. Формирование начального набора образцов

На первом шаге работы алгоритма ЛП-ДА необходимо задать начальный набор образцов  $\mathfrak{P} = \{A, B\}$ . Очевидно, что множества  $A$  и  $B$  должны быть не пусты. Однако это не является достаточным условием для обеспечения эффективной работы алгоритма. Рассмотрим пример, изображенный на Рис. 8. Здесь  $D = \{x \mid Ax \leq b\}$  - допустимое множество точек формализованной части задачи ЛПНО;  $\bar{P} = \{x \mid \bar{\varphi}(x) \leq 0\}$  - полупространство, определяемое неформализованным ограничением  $\bar{\varphi}(x) \leq 0$ ;  $\tilde{P} = \{x \mid \tilde{\varphi}(x) \leq 0\}$  - полупространство, задаваемое разделяющей функцией  $\tilde{\varphi}(x)$ . Мы видим, что в данном случае  $\tilde{\varphi}$  является допустимой разделяющей функцией для множеств  $A$  и  $B$ . Тем не менее, аппроксимационная задача (2.1) в этом случае является некорректной (не имеет решения).

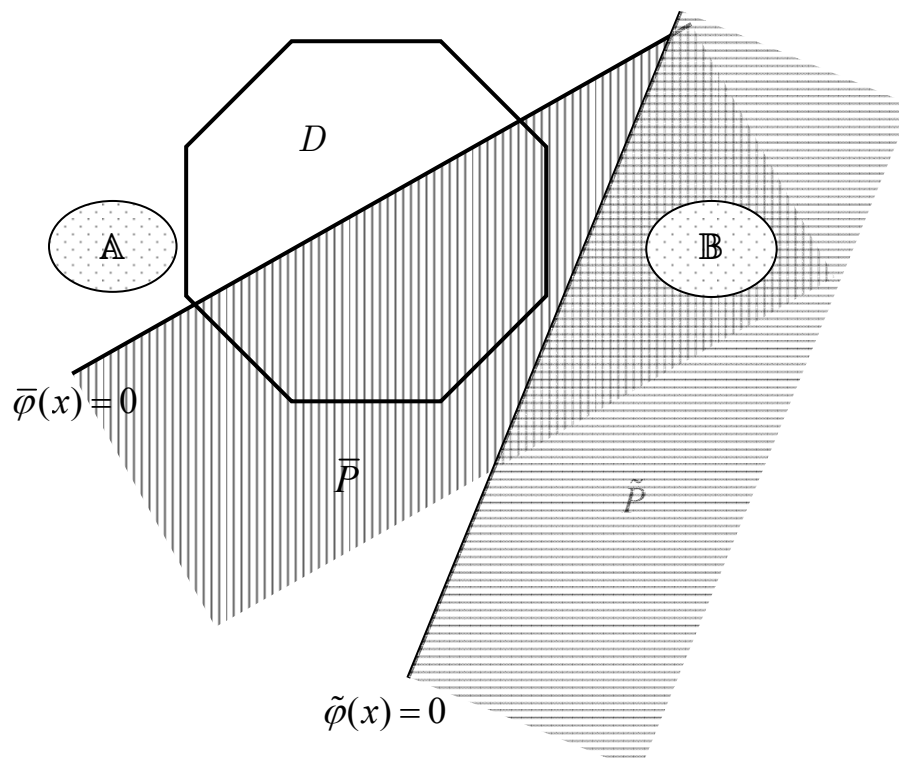


Рис. 8. Некорректный начальный набор образцов.

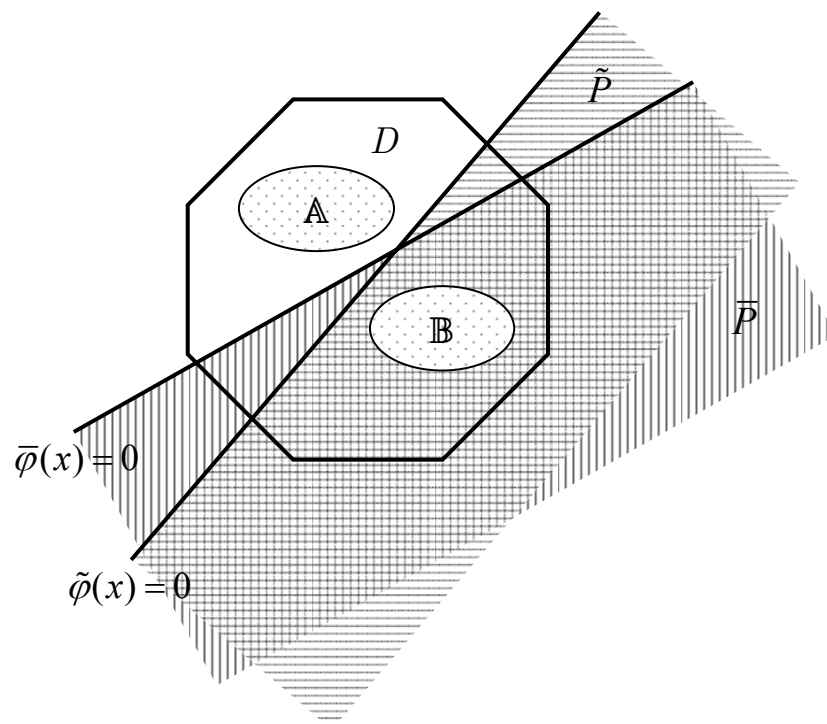


Рис. 9. Корректный начальный набор образцов.

Очевидно, что эффективную работу алгоритма ЛП-ДА обеспечивает набор образцов  $\mathfrak{P} = \{A, B\}$ , удовлетворяющий следующим условиям (Рис. 9):

$$\begin{aligned} A \subset D \setminus \bar{P}, \quad B \subset D \cap (\bar{P} \setminus \bar{H}), \\ A \neq \emptyset, \quad B \neq \emptyset. \end{aligned} \quad (2.2)$$

Здесь  $\bar{H} = \{x \mid \bar{\varphi}(x) = 0\}$  – гиперплоскость, определяемая неформализованным ограничением. При этом мы предполагаем, что множества  $D \setminus \bar{P}$  и  $D \cap (\bar{P} \setminus \bar{H})$  имеют внутренние точки. В этом случае аппроксимационная задача (2.1) всегда будет иметь решение.

В соответствии с этим можно использовать следующий алгоритм для формирования начального набора образцов. Мы находим все вершины многогранника  $D$ , применяем к ним функцию эксперта  $\epsilon$  и формируем множества  $A$  и  $B$  по следующему правилу:

$$\mathbb{A} = \{x \mid \mathbf{e}(x) = 1\},$$

$$\mathbb{B} = \{x \mid \mathbf{e}(x) = -1\}.$$

Точки, для которых  $\mathbf{e}(x) = 0$ , отбрасываются. Если множества  $D \setminus \bar{P}$  и  $D \cap (\bar{P} \setminus \bar{H})$  имеют внутренние точки, то в результате мы получим набор образцов, удовлетворяющий условиям (2.2).

### 2.3. Критерий завершения итерационного процесса

Для реализации алгоритма ЛП-ДА в виде программы нам необходим некоторый критерий близости решения аппроксимационной задачи к точному решению задачи ЛПНО. Этот критерий необходим для завершения итерационного процесса.

Пусть  $\tilde{x}^0, \tilde{x}^1, \dots, \tilde{x}^k$  – последовательность решений аппроксимационных задач, полученных при работе алгоритма ЛП-ДА. Тогда в качестве искомого критерия может выступать условие вида

$$\sum_{i=1}^l \|\tilde{x}^k - \tilde{x}^{k-i}\| < \varepsilon,$$

где  $\varepsilon > 0$  – заданное вещественное число, определяющее точность вычислений. Здесь  $l$  – фиксированное целое положительное число, являющееся параметром алгоритма ЛП-ДА.

Корректность предложенного критерия обеспечивается следующей теоремой.

**Теорема 6.** Пусть  $\bar{x}$  – решение задачи ЛПНО,  $\{\tilde{x}^k\}_{k=0}^{\infty}$  – последовательность решений аппроксимационных задач, порождаемых алгоритмом ЛП-ДА,  $l$  – фиксированное целое положительное число. Тогда

$$\lim_{k \rightarrow \infty} \|\bar{x} - \tilde{x}^k\| = 0 \quad \Rightarrow \quad \lim_{k \rightarrow \infty} \sum_{i=1}^l \|\tilde{x}^k - \tilde{x}^{k-i}\| = 0.$$

**Доказательство.** Пусть  $\lim_{k \rightarrow \infty} \|\bar{x} - \tilde{x}^k\| = 0$ . Это означает, что

$$\forall \varepsilon > 0 : \exists k' : \forall k > k' : \|\bar{x} - \tilde{x}^k\| < \varepsilon.$$

Непосредственно отсюда следует, что

$$\forall \varepsilon > 0 : \exists k' : \forall k > k' : (\|\bar{x} - \tilde{x}^k\| < \varepsilon \ \& \ \|\bar{x} - \tilde{x}^{k+i}\| < \varepsilon).$$

То есть

$$\forall \varepsilon > 0 : \exists k' : \forall k > k' : \|\tilde{x}^k - \tilde{x}^{k+i}\| < 2\varepsilon,$$

откуда получаем  $\lim_{k \rightarrow \infty} \|\tilde{x}^k - \tilde{x}^{k+i}\| = 0$ . Это равносильно

$\lim_{k \rightarrow \infty} \|\tilde{x}^k - \tilde{x}^{k-i}\| = 0$ , откуда следует

$$\lim_{k \rightarrow \infty} \sum_{i=1}^l \|\tilde{x}^k - \tilde{x}^{k-i}\| = 0.$$

Теорема доказана.

При реализации алгоритма ЛП-ДА необходимо выбирать значения параметра  $l$  большими единицы, так как близость только одной пары точек может оказаться случайностью. С другой стороны, не следует брать значения параметра  $l$  слишком большими, так как это может существенно замедлить работу программы и, более того, привести к "зацикливанию" вычислительного процесса в результате накопления ошибок, связанных с погрешностью вычислений. Проведенные вычислительные эксперименты на модельных и реальных задачах ЛПНО показали, что в подавляющем большинстве случаев  $l = 2$  дает хорошие результаты.

## 2.4. Рандомизация

На каждой итерации алгоритм ЛП-ДА строит очередную аппроксимационную задачу. Точка  $\tilde{x}$ , полученная в качестве решения данной аппроксимационной задачи, используется для пополнения набора образцов, в результате чего формируется следующая аппроксимационная задача. При этом могут возникнуть коллизии трех типов. Коллизия первого типа возникает, когда  $\tilde{x}$  принадлежит гиперплоскости  $\bar{H}$ , задаваемой неформализо-

ванным ограничением. В этом случае точка  $\tilde{x}$  не может быть отнесена ни к множеству  $\mathbb{A}$ , ни к множеству  $\mathbb{B}$ . Коллизии второго и третьего типов возникают, когда  $\tilde{x}$  уже присутствует в множествах  $\mathbb{A}$  или  $\mathbb{B}$ . Для разрешения этих коллизий мы используем следующую *процедуру рандомизации*.

Определим *случайное* отображение  $\mathbf{r}^\rho : \mathbb{R}^n \rightarrow \mathbb{R}^n$ , обладающее свойством:  $\|\mathbf{r}^\rho(x) - x\| < \rho$ . Отображение  $\mathbf{r}^\rho(x)$  порождает случайные точки внутри сферы радиуса  $\rho$  с центром в  $x$ . Вещественное число  $\rho > 0$  является параметром алгоритма ЛП-ДА, задающим *радиус рандомизации*. На языке Си отображение  $\mathbf{r}^\rho(x)$  реализуется путем применения генератора псевдослучайных чисел **random** к каждой координате точки  $x$ . Процедура рандомизации заключается в построении с помощью отображения  $\mathbf{r}^\rho(\tilde{x})$  множества точек  $\mathfrak{R}_m^\rho = \{\hat{x}^1, \dots, \hat{x}^m\}$ , удовлетворяющих условию

$$\hat{x}^i \notin \mathbb{A} \cup \mathbb{B} \cup \bar{H}, \quad \|\hat{x}^i - \tilde{x}\| < \rho \quad (i = 1, \dots, m).$$

Целое положительное число  $m$  является параметром алгоритма ЛП-ДА, задающим *мощность рандомизации*. Все точки множества  $\mathfrak{R}_m^\rho$  подвергаются экспертизе и используются для пополнения набора образцов на текущей итерации.

Очевидно, что увеличение мощности рандомизации  $m$  должно уменьшать количество итераций метода ЛП-ДА, необходимое для получения приближенного решения с заданной точностью. Однако при этом одновременно будет увеличиваться количество экспертиз, выполняемых в пределах одной итерации.

Увеличение радиуса рандомизации  $\rho$  на начальных итерациях также должно приводить к ускорению сходимости. В то же время на более поздних итерациях большой радиус рандомизации может заметно уменьшить скорость сходимости, так как рандомизированные точки будут порождаться далеко от точного решения, что, очевидно, приведет к ухудшению эффективности пополнения набора образцов.

Проблема выбора оптимальных значений параметров  $\rho$  и  $m$  при решении практических задач с помощью алгоритма ЛП-ДА будет исследована в разделах 4.1.2 и 4.1.3 соответственно.

## 2.5. Метод осцилляций

При применении метода ЛП-ДА к решению практических задач мы можем столкнуться с ограничением на максимально допустимое количество итераций. Данное ограничение может быть обусловлено, например, ограничениями на максимальное количество обращений к эксперту или на максимальное время нахождения решения. Для сокращения количества итераций и, соответственно, для ускорения сходимости метода ЛП-ДА было предложено усовершенствование исходного алгоритма, получившее название *метод осцилляций* [39].

Суть метода осцилляций состоит в том, что на каждой итерации алгоритма ЛП-ДА кроме аппроксимационной задачи (2.1) решается следующая *дополнительная* задача линейного программирования (ЛПД):

$$\tilde{x} \in \text{Arg min} \{ (c, x) \mid Ax \leq b, \tilde{\varphi}(x) \geq 0 \} \quad (2.3)$$

(см. Рис. 10).

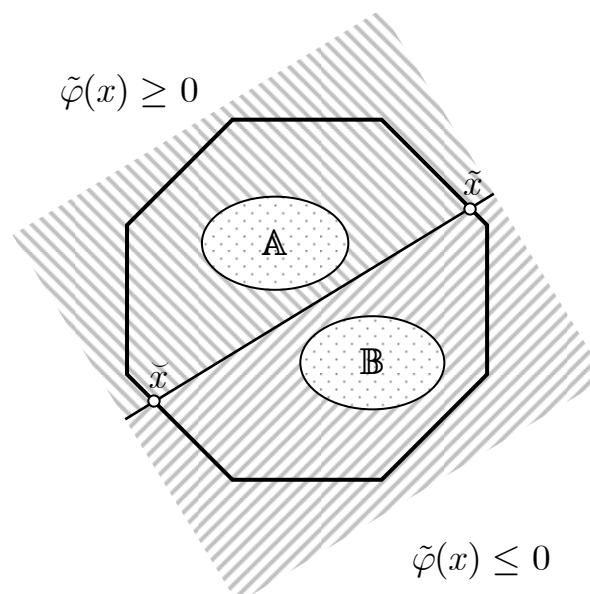


Рис. 10. Метод осцилляций.

Решение  $\tilde{x}$  дополнительной задачи подвергается процедуре экспертизы и, в случае отсутствия коллизий, добавляется к набору образцов  $\mathfrak{R} = \{A, B\}$ . При возникновении коллизии к точке  $\tilde{x}$  применяется процедура рандомизации. Мы здесь предполагаем для простоты, что аппроксимационная и дополнительная задачи имеют единственное решение.

Метод осцилляций может значительно ускорить сходимость алгоритма ЛП-ДА на начальных итерациях, когда набор образцов не является еще достаточно представительным. Это было подтверждено проведенными вычислительными экспериментами (см. раздел 4.1.4). Однако на завершающем этапе вычислений метод осцилляций теряет свою эффективность и может привести к замедлению итерационного процесса. Следовательно, при реализации алгоритма ЛП-ДА необходимо на определенном этапе итерационного процесса отключать процедуру осцилляции. С этой целью был введен параметр  $\lambda$ , задающий максимально допустимое количество последовательных итераций, на которых возникли коллизии с решением  $\tilde{x}$  дополнительной задачи. Если количество последовательных коллизий  $\tilde{x}$  превысило  $\lambda$ , то процедура осцилляции отключается до завершения итерационного процесса. Проведенные вычислительные эксперименты на реальных и искусственных задачах показали, что в большинстве случаев значение  $\lambda = 2$  является хорошим выбором.

## **2.6. Проблема погрешности вычислений**

Известно, что задача линейного программирования является устойчивой по всей совокупности данных, если оптимальные множества исходной и двойственной задач – ограничены [12]. Однако, при использовании симплекс-метода для линейной оптимизации мы столкнемся с проблемой его неустойчивости по отношению к нулевым значениям. Эта проблема заключается в том, что если некоторое нулевое значение, полученное в ходе выполнения итераций симплекс-метода, заменить на сколь угодно малое не-

нулевое значение, то это может привести к тому, что либо полученное решение не будет являться решением исходной задачи, либо исходная разрешимая задача превратится в неразрешимую. Замена нулевого значения на ненулевое происходит в компьютерных вычислениях в результате неточного представления вещественных чисел в ячейках памяти. В соответствии с этим при реализации симплекс-метода мы использовали следующие допуски на ошибки округления, применяемые вместо точной проверки на нуль.

$\delta_{aij}$  – допуск для элементов  $a_{ij}$  симплекс-таблицы: если после выполнения очередной итерации абсолютное значение элемента  $a_{ij}$  симплекс-таблицы становится меньше, чем  $\delta_{aij}$ , то необходимо  $a_{ij}$  присвоить нуль.

$\delta_{col}$  – допуск для выбора разрешающего столбца: при выборе разрешающего столбца не рассматриваются те элементы, значения которых по абсолютной величине меньше, чем  $\delta_{col}$ .

$\delta_{row}$  – допуск для выбора разрешающей строки: при выборе разрешающей строки не рассматриваются те элементы, значения которых меньше, чем  $\delta_{row}$ .

В Табл. 1 приведены величины указанных допусков в зависимости от разрядности машинного слова. Это типичные значения для представления чисел в формате с плавающей точкой [31]. При выборе указанных значений мы исходим из того, что 32-разрядное слово позволяет хранить числа примерно с девятью десятичными значащими цифрами, а 64-разрядное – с семнадцатью. В языке Си 32-разрядному представлению обычно соответствует тип **float**, а 64-разрядному представлению – тип **double**. Различные величины допусков для одной разрядности отражают относительную важность допустимой ошибки.

**Табл. 1.** Величины допусков для симплекс-метода.

Допуск	Число разрядов в машинном слове		
	32	60	64
$\delta_{aij}$	$10^{-6}$	$10^{-10}$	$10^{-11}$
$\delta_{col}$	$10^{-5}$	$10^{-8}$	$10^{-9}$
$\delta_{row}$	$10^{-3}$	$10^{-5}$	$10^{-6}$

В отличие от симплекс-метода, метод ЛП-ДА требует введения еще трех дополнительных допусков:

$\delta_{\text{ЛПА}}$  – допуск для решения  $\tilde{x}$  аппроксимационной задачи (2.1): если после выполнения очередной итерации метода ЛП-ДА  $|\lceil \tilde{x}_j \rceil - \tilde{x}_j| < \delta_{\text{ЛПА}}$ , то необходимо  $\tilde{x}_j$  присвоить значение  $\lceil \tilde{x}_j \rceil^1$  ( $j = 1, \dots, n$ ).

$\delta_{\text{ЛПД}}$  – допуск для решения  $\check{x}$  дополнительной задачи (2.3): если после выполнения очередной итерации метода ЛП-ДА  $|\lceil \check{x}_j \rceil - \check{x}_j| < \delta_{\text{ЛПД}}$ , то необходимо  $\check{x}_j$  присвоить значение  $\lceil \check{x}_j \rceil$  ( $j = 1, \dots, n$ ).

$\delta_{\bar{H}}$  – допуск для попадания  $\tilde{x}$  на гиперплоскость  $H$ , задаваемую неформализованным ограничением: если  $\text{dist}(\tilde{x}, \bar{H}) < \delta_{\bar{H}}^2$ , то к  $\tilde{x}$  необходимо применить процедуру рандомизации.

**Табл. 2.** Величины дополнительных допусков для метода ЛП-ДА.

Допуск	Число разрядов в машинном слове		
	32	60	64
$\delta_{\text{ЛПА}}$	$10^{-2}$	$10^{-3}$	$10^{-4}$
$\delta_{\text{ЛПД}}$	$10^{-1}$	$10^{-2}$	$10^{-2}$
$\delta_{\bar{H}}$	$10^{-3}$	$10^{-5}$	$10^{-6}$

<sup>1)</sup> Здесь  $\lceil x \rceil$  обозначает число, получающееся из  $x$  путем его округления до целого.

<sup>2)</sup> Здесь  $\text{dist}(\tilde{x}, \bar{H}) = \inf \{ \|\tilde{x} - h\| : h \in \bar{H} \}$ .

Последний допуск  $\delta_{\bar{H}}$  должен использоваться экспертом при квалификации новых образцов. В Табл. 2 приведены значения дополнительных допусков в зависимости от разрядности машинного слова. Данные значения были получены экспериментальным путем.

Механизм допусков хорошо работает для задач ЛП, представленных *большими числами* (значительно превосходящими по абсолютной величине значения допусков). Однако, если задача ЛП представлена *малыми числами* (сравнимыми с величинами допусков), то указанный прием не срабатывает, так как в результате погрешностей вычислений мы перестаем различать ненулевые и нулевые значения. В этом случае можно использовать модифицированный симплекс-метод с *LU-декомпозицией* [47], являющийся более стабильным по отношению к нулевым значениям (но и более медленным по сравнению со стандартным симплекс-методом). Другим решением проблемы малых чисел является *масштабирование* исходной задачи [77].

Эксперименты, проведенные на реальных и искусственных задачах ЛП, показали, что подход, основанный на использовании допусков, приведенных в Табл. 1 и Табл. 2, обеспечивает высокую стабильность алгоритма ЛП-ДА по отношению к нулевым значениям.

## ГЛАВА 3. ПРОГРАММНЫЙ КОМПЛЕКС ЛП-ДА

На основе описанной методики был спроектирован программный комплекс для решения задач линейного программирования с неформализованными ограничениями, соединяющий в себе методы линейного программирования и дискриминантного анализа.

### 3.1. Модульная структура комплекса

Иерархическая модульная структура программного комплекса изображена на Рис. 11. Программный комплекс включает в себя следующие модули:

- Головной модуль;
- Модуль универсального алгоритма;
- Модуль экспертизы;
- Модуль оптимизации;
- Модуль дискриминации.

*Головной модуль* обеспечивает общее управление вычислительным процессом.

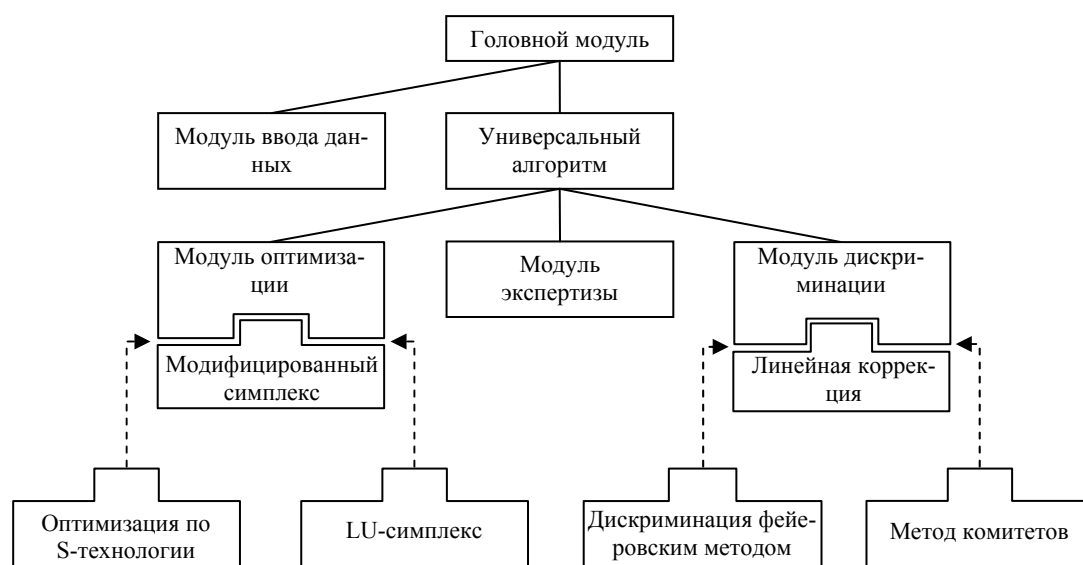


Рис. 11. Модульная структура программного комплекса.

*Модуль ввода данных* выполняет ввод исходных данных задачи ЛПНО из текстового файла. Входной файл данных задачи ЛПНО представляется в *формате LPNC*, разработанном специально для представления задач линейного программирования с неформализованными ограничениями. Формат LPNC является расширением промышленного стандарта *MPS* [31, 72], который используется для подготовки данных в большинстве коммерческих систем линейного программирования. Детально формат LPNC будет описан в разделе 3.2.

*Модуль универсального алгоритма* реализует общий алгоритм решения задачи ЛПНО, изображенный на Рис. 7. На фазе оптимизации модуль универсального алгоритма вызывает *метод оптимизации* из *модуля оптимизации*. Реализация метода оптимизации скрыта в модуле оптимизации. На фазе дискриминантного анализа модуль универсального алгоритма вызывает *метод дискриминантного анализа* из *модуля дискриминации*. Реализация этого метода скрыта в модуле дискриминации.

*Модуль экспертизы* реализует функцию эксперта. В качестве конкретной реализации модуля экспертизы может фигурировать интерфейс, обеспечивающий связь с *внешним экспертом*, нейросетевая программа или экспертная система.

*Модуль оптимизации* реализует метод линейной оптимизации. Программный комплекс предусматривает возможность использования следующих методов линейного программирования:

- 1) Симплекс-метод с LU-разложением [47];
- 2) Модифицированный симплекс-метод [7, 52, 73];
- 3) Метод линейной оптимизации на базе фейеровских приближений (S-технология) [9, 13].

Данные алгоритмы реализуются в программном комплексе в виде *сменных картриджей*, которые подключаются к модулю оптимизации на этапе компиляции.

*Модуль дискриминации* реализует метод дискриминантного анализа. Программный комплекс предусматривает возможность использования следующих алгоритмов дискриминантного анализа:

- 1) Линейная коррекция [17, 33];
- 2) Метод комитетов [26];
- 3) Фейеровские методы сильной отделимости [13].

Указанные алгоритмы также реализуются в виде сменных картриджей, которые подключаются к модулю дискриминации на этапе компиляции.

Технология картриджей позволяет сочетать различные комбинации алгоритмов, что дает возможность строить конфигурации программного комплекса, наилучшим образом учитывающие специфику каждой конкретной задачи.

### **3.2. Формат входных данных LPNC**

В качестве файла исходных данных программный комплекс использует текстовый файл, содержащий описание задачи ЛПНО в *формате LPNC (Linear Programming with Nonformalized Constraints)*. Формат LPNC разработан на базе формата *MPS (Mathematical Programming System)* [31, 72], который является промышленным стандартом для большинства коммерческих программных систем, ориентированных на решение задач математического программирования.

Для каждого образца в формате LPNC вводится уникальное имя (идентификатор), которое вместе с типом образца должно быть указано в секции ROWS. Тип образца записывается в позициях 2-3, а имя образца – в позициях 5-12. Для спецификации типа образца формат LPNC предусматривает два дополнительных кода, приведенных в Табл. 3. Координаты каждого образца задаются в секции COLUMNS таким же образом, как и коэффициенты матрицы формализованных ограничений.

Табл. 3. Коды для спецификации образцов.

Код	Тип образца	Семантика
A	$x \in \mathbb{A}$	Образец не удовлетворяет неформализованным ограничениям
B	$x \in \mathbb{B}$	Образец удовлетворяет неформализованным ограничениям

При исследовании алгоритмов решения задач ЛПНО иногда полезно моделировать эксперта путем явного задания неформализованных ограничений. Для этих целей в формате LPNC предусмотрены специальные NFC-строки. Данные строки не описываются в секции ROWS LPNC-файла входных данных. Вместо этого в секции COLUMNS выделяются переменные, образующие NFC-множества. Карты-маркеры, которые указывают на начало и конец группы столбцов, принадлежащих отдельному NFC-множеству, имеют формат, приведенный в Табл. 4.

Табл. 4. Формат NFC-множества

	Поле 1 (2-3)	Поле 2 (5-12)	Поле 3 (15-22)	Поле 4 (25-36)	Поле 5 (40-47)	Поле 6 (50-61)
Маркер начала	G, L или E	<Имя NFC-строки>	'MARKER'		'NFCORG'	
Карты данных		<Имя столбца>	<Имя строки>	<Значение>	<Имя строки>	<Значение>
Маркер конца		<Имя NFC-строки>	'MARKER'		'NFCEND'	

Карты данных имеют тот же формат, который определяется стандартом MPS. В них задаются ненулевые элементы, принадлежащие как обычным строкам, так и NFC-строкам. NFC-множества, соответствующие различным неформализованным ограничениям, могут между собой пересекаться.

Компоненты вектора ограничений каждой NFC-строки специфицируются в секции RHS, как если бы это были обычные строки. При этом в качестве имени строки в поле 3 или 5 указывается имя соответствующей NFC-строки.

Рассмотрим следующий пример задачи ЛПНО. Пусть формализованные ограничения задаются системой линейных неравенств:

$$\begin{cases} x_1 + x_2 \leq 3 \\ x_1 - x_2 \leq 1 \\ x_1 - 2x_2 \geq -1 \end{cases}$$

Пусть имеются два неформализованных ограничения:

$$\begin{cases} -1.25x_1 + 3x_2 \leq 3.75 \\ x_1 \geq 2.5 \end{cases}$$

Предположим, что целевая функция имеет вид  $Q_{\max} = x_2$ . Пусть набор образцов задается множествами  $\mathbb{A} = \{(1;2)\}$ ,  $\mathbb{B} = \{(2;1), (1;0), (0;1)\}$ . Тогда входной файл задачи будет иметь вид, изображенный на Рис. 12.

```

*          1          2          3          4          5          6
*23456789012345678901234567890123456789012345678901234567890
NAME          example
ROWS
L  ue_1
L  ue_2
G  ue_3
N  Qmax
A  pa_1
B  pb_1
B  pb_2
B  pb_3
COLUMNS
L  nc_1          'MARKER'          'NFCORG'
G  nc_2          'MARKER'          'NFCORG'
   x_1          ue_1          1.0          ue_2          1.0
   x_1          ue_3          1.0          nc_1          -1.25
   x_1          nc_2          1.0          pa_1          1.0
   x_1          pb_1          2.0          pb_2          1.0
   nc_2          'MARKER'          'NFCEND'
   x_2          ue_1          1.0          ue_2          -1.0
   x_2          ue_3          -2.0          nc_1          3.0
   x_2          pa_1          2.0          pb_1          1.0
   x_2          pb_3          1.0          Qmax          1.0
   nc_1          'MARKER'          'NFCEND'
RHS
   b          ue_1          3.0          ue_2          1.0
   b          ue_3          -1.0          nc_1          3.75
   b          nc_2          2.5
ENDATA

```

Рис. 12. LPNC-файл для задачи с неформализованным ограничением.

### 3.3. Параллельная реализация

В практике экономико-математического моделирования часто встречаются задачи с большим количеством переменных и ограничений. Размер типичной средней задачи может составлять 20 000 переменных и 5 000 ограничений [61]. В отдельных случаях количество переменных может превышать 100 000, а количество ограничений – 20 000. Подобные задачи линейного программирования при решении требуют значительных вычислительных мощностей. В случае наличия неформализованных ограничений объем требуемых вычислений может возрасти многократно. В связи с этим актуальной является задача разработки параллельных алгоритмов решения задачи ЛПНО.

#### 3.3.1. Классификация параллельных методов решения задачи ЛПНО

При анализе структуры алгоритма ЛП-ДА мы можем выделить следующие классы методов параллельного решения задачи ЛПНО (Рис. 13).

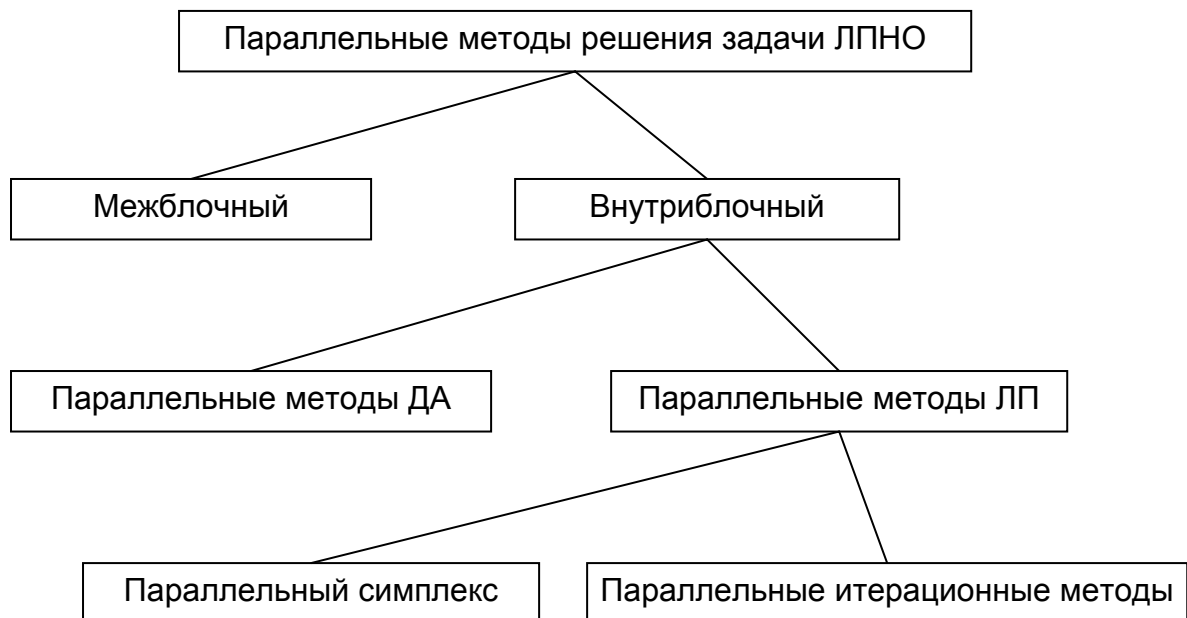


Рис. 13. Классификация параллельных методов решения задачи ЛПНО.

*Межблочный метод* предполагает разбиение задачи ЛПНО на блоки, каждый из которых параллельно вычисляет одно неформализованное ограничение, после чего симплекс-методом вычисляется приближенное решение исходной задачи ЛПНО так, как это было описано в разделе 1.7. Детально межблочный метод будет описан в разделе 3.3.2.

*Внутриблочный метод* предполагает параллельное выполнение отдельных блоков алгоритма ЛП-ДА (см. Рис. 7 на стр. 36). В соответствии с этим мы можем рассматривать параллельные методы дискриминантного анализа (блок дискриминантного анализа) и параллельные методы линейного программирования (блок оптимизации).

*Параллельные методы дискриминантного анализа* в подавляющем большинстве случаев сводятся к использованию параллельных нейросетевых алгоритмов [53, 65]. Метод линейной коррекции, используемый в блоке дискриминантного анализа, не поддается эффективному распараллеливанию. Более перспективными в этом отношении являются фейеровские методы сильной отделимости выпуклых полиэдральных множеств [13].

*Параллельные методы линейного программирования* можно разбить на две основные группы: параллельный симплекс-метод и параллельные итерационные методы. *Симплекс-метод* в общем случае допускает эффективное распараллеливание в том случае, когда число процессоров не превышает 30 [76]. Однако, в случае, когда матрица  $A$  имеет специальную блочно-диагональную структуру, эффективного распараллеливания можно добиться и на большем количестве процессоров [67]. В основе параллельного алгоритма в этом случае лежит принцип декомпозиции Данцига-Вульфа [49, 68, 70].

*Параллельные итерационные методы* решения задачи линейного программирования базируются главным образом на S-технологии [9, 13] в основе которой лежит использование фейеровских отображений. Здесь наметились два подхода. Первый подход ориентирован на матрицы со специ-

альной блочной структурой [1, 2, 12]. Преимуществом этого подхода является отсутствие массовых пересылок данных между процессорами. Главным недостатком является то, что степень параллелизма при использовании данного подхода ограничена количеством блоков, на которые разбивается матрица  $A$ . Второй подход [42] основан на разбиении симметрической задачи на подзадачи путем проектирования допустимого множества на гиперплоскости, задаваемые координатными осями. Итерационный процесс выполняется параллельно для каждой проекции. Через определенное число шагов происходит периодическое *связывание* независимых итерационных процессов путем подстановки данных в общую симметрическую задачу. Однако этот подход нуждается в дальнейших исследованиях.

### 3.3.2. Параллельная версия алгоритма ЛП-ДА

Схема параллельного алгоритма решения задачи ЛПНО\*, основанного на межблочном методе, приведена на Рис. 14. На первом шаге в соответствии с подходом, описанным в 1.7, задача ЛПНО\*, имеющая  $r$  неформализованных ограничений, разбивается на  $r$  задач ЛПНО<sub>1</sub>, ..., ЛПНО <sub>$r$</sub> . Каждая из этих задач независимо решается методом ЛП-ДА. Из вычисленных приближений неформализованных ограничений и формализованной части задачи ЛПНО строится полная система ограничений.

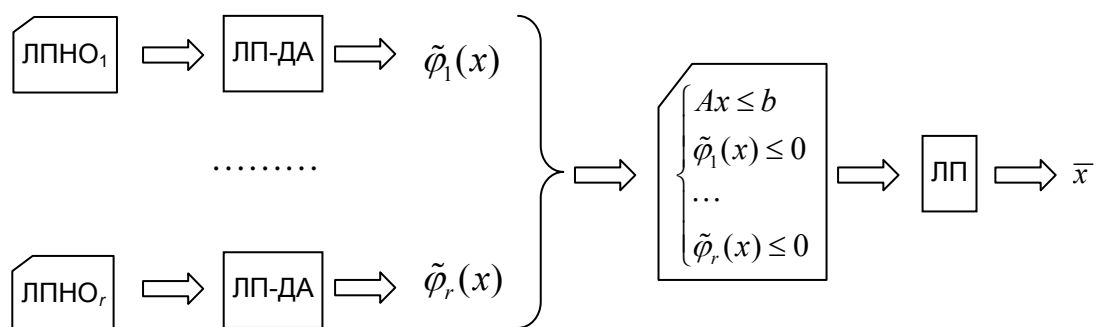


Рис. 14. Схема параллельного алгоритма.

На втором шаге полученная таким образом задача решается одним из методов линейного программирования. В результате с некоторым приближением мы получаем  $\bar{x}$  – решение исходной задачи ЛПНО\*.

При реализации параллельной версии алгоритма ЛП-ДА нахождение каждого неформализованного ограничения оформлялось в виде независимого процесса [43]. Вычисленные ограничения передавались некоторому выделенному процессору-координатору, который формировал полную задачу линейного программирования и решал ее каким-либо методом линейной оптимизации. Указанный подход позволяет без какой-либо модификации выполнять параллельную программу на различном количестве процессоров. На Рис. 15 изображены примеры распределения процессов по процессорам: а) на каждом процессоре выполняется один процесс решения задачи ЛПНО; б) на каждом процессоре выполняется одновременно два процесса решения задач ЛПНО. В общем случае на одном процессоре может выполняться произвольное число процессов ЛПНО. В качестве процессора-координатора во всех случаях выступает процессор с номером 1.

Для организации обменов данными между процессами была использована система параллельного программирования MPI, основанная на передаче сообщений [34, 50, 75].

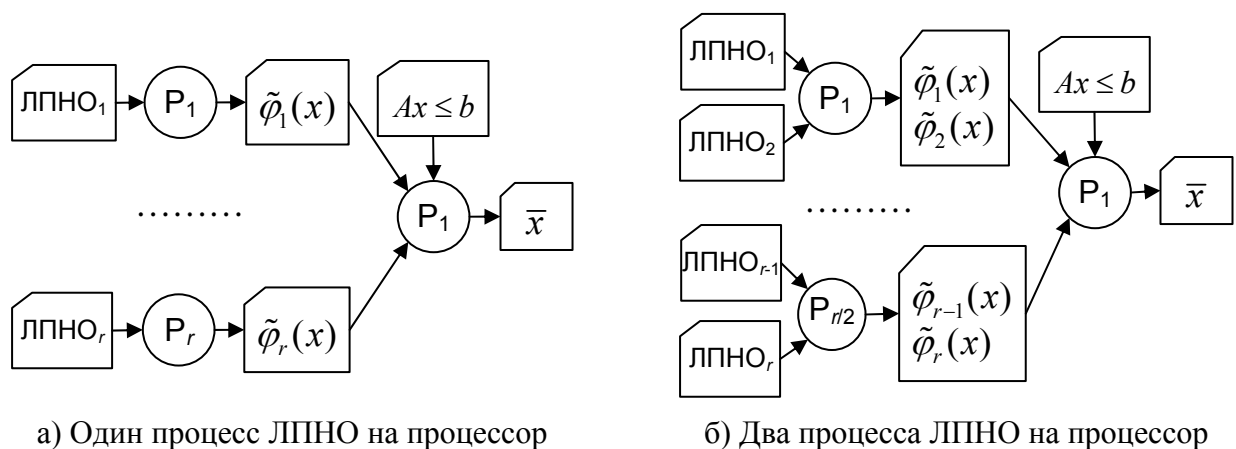


Рис. 15. Распределение процессов ЛПНО по процессорам.

Стандарт MPI в настоящее время является наиболее распространенной средой параллельного программирования для многопроцессорных систем с массовым параллелизмом [5, 19]. Интерфейс MPI поддерживает создание параллельных программ в стиле MIMD [4, 60], что подразумевает объединение в одну программу процессов с различными исходными текстами. Однако на практике гораздо чаще используют SIMD-модель [4, 60], в рамках которой для всех параллельных процессов используется один и тот же код. Практически все современные реализации MPI поддерживают работу с нитями. Это позволяет эффективно использовать технологию MPI на многопроцессорных системах с разделяемой памятью и на системах с многоядерной архитектурой.

В 2002 г. появилась реализация нового стандарта MPI-2 [56, 57]. Отличительной особенностью стандарта MPI-2 является наличие функций PUT и GET, позволяющих организовать так называемые *односторонние коммуникации*. Процесс-поставщик и процесс-потребитель осуществляют операции записи и чтения данных независимо друг от друга через общесистемный распределенный буфер. Синхронизация процессов осуществляется посредством функции FENCE.

При реализации параллельной версии алгоритма ЛП-ДА мы использовали передачу сообщений на основе односторонних коммуникаций MPI-2. Проектирование программы выполнялось в соответствии с моделью SIMD. Схема реализации параллельного алгоритма на псевдокоде в виде процедуры LPDA изображена на Рис. 16. Приведенная реализация адаптирована для любого количества неформализованных ограничений и любого количества процессоров. Процедура LPDA запускается как процесс MPI. Количество запущенных процессов MPI задается переменной среды `MPI_Community_size`. Номер процесса MPI, который выполняет данный экземпляр процедуры LPDA, хранится в переменной среды `MPI_Community_World_rank`.

```

// Параллельный алгоритм ЛП-ДА
procedure LPDA begin
  myId = MPI_Community_World_rank; // номер процессора
  if myId > r then // номер процессора превышает кол-во неформализ. огр-й
    exit procedure;
  end if;
  numProcs = MPI_Community_Size; // количество процессоров
  for i=myId to r step numProcs begin // цикл по неформализов. огр-ям
    loop begin // итерационный цикл вычисления неформализов. ограничения
       $\phi_i = DA(\mathfrak{P});$  // дискрим. анализ: находим i-тую разделяющую функцию
      ЛПА = ЛПФ  $\oplus \phi_i(x) \leq 0$  // формируем задачу ЛПА
       $x = LP(\text{ЛПА});$  // решаем задачу ЛПА
       $\mathfrak{P} = \mathfrak{P} \oplus x;$  // добавляем  $x$  к набору образцов после экспертизы
      if  $\varepsilon(x)$  then // если выполняется критерий завершения
        exit loop; // выходим из итерационного цикла
      end if;
      ЛПД =  $\ominus$ ЛПФ  $\oplus \phi_i(x) \geq 0$  // формируем задачу ЛПД
       $y = LP(\text{ЛПД});$  // решаем задачу ЛПД
       $\mathfrak{P} = \mathfrak{P} \oplus y;$  // добавляем  $y$  к набору образцов после экспертизы
    end loop;
    MPI_Put( $\phi_i, 1$ ); // пересылаем  $\phi_i$  на процессор с номером 1
    MPI_Win_fence(); // синхронизация процессоров
  end for;
  if myId == 1 then // только для процесса с номером 1
    ЛП = ЛПФ; // формализованная часть задачи ЛПНО*
    for i=1 to r begin
      ЛП = ЛП  $\oplus \phi_i(x) \leq 0$  // добавляем неформал. огр-я к формал. части
    end for;
     $x = LP(\text{ЛПА});$  // решаем задачу линейного программирования
    print  $x;$  // печатаем результат
  end if;
  exit procedure;
end procedure;

```

Рис. 16. Схема параллельного алгоритма ЛП-ДА.

Каждый MPI процесс  $i$  вычисляет свою группу неформализованных ограничений, начиная с ограничения с номером  $i$ . Если количество неформализованных ограничений меньше либо равно количеству процессов, то каждый процесс будет вычислять не более одного неформализованного огра-

ничения. Если количество неформализованных ограничений больше количества процессов, то каждый процесс будет вычислять не более  $\lceil r/k \rceil$  неформализованных ограничений (здесь  $r$  – количество неформализованных ограничений,  $k$  – количество процессов MPI). Все найденные неформализованные ограничения передаются на процесс с номером 1, после чего происходит синхронизация процессов. Процесс с номером 1 добавляет все полученные от других процессов ограничения к формализованной части задачи ЛПНО\* и решает получившуюся задачу линейного программирования каким-либо методом линейной оптимизации.

Предложенная параллельная версия алгоритма ЛП-ДА может использоваться без каких-либо изменений как на многопроцессорных системах с общей памятью, так и на системах с массовым параллелизмом, включая кластерные системы. При этом распределение процессов MPI по физическим процессорам выполняется операционной системой. Масштабируемость описанной параллельной версии алгоритма ЛП-ДА будет исследована в разделе 4.3.

### **3.4. Реализация прототипа**

В рамках диссертационного исследования была выполнена реализация прототипа программного комплекса ЛП-ДА на языке Си. В качестве картриджа для блока оптимизации был реализован симплекс-метод, в качестве картриджа для блока дискриминантного анализа был реализован метод линейной коррекции. Работа эксперта моделировалась явным заданием неформализованных ограничений. Общий объем кода на языке Си составил около 3000 строк. Исходные тексты прототипа свободно доступны в Интернет по адресу: <http://life.susu.ru/lpno/>.

Кроме этого была также реализована параллельная версия алгоритма ЛП-ДА с использованием пакета MPI-2, описанная в разделе 3.3.2. Исход-

ные тексты параллельной версии свободно доступны в Интернет по адресу:  
<http://life.susu.ru/lpnoMPI/>.

## ГЛАВА 4. КОМПЬЮТЕРНЫЙ АНАЛИЗ АЛГОРИТМА ЛП-ДА

Для исследования влияния различных параметров на время работы и количество итераций алгоритма ЛП-ДА была проведена серия вычислительных экспериментов на базе разработанного программного комплекса. В качестве тестовых задач были использованы как реальные, так и искусственные задачи. Искусственные задачи дают хорошую возможность для подбора оптимальных значений параметров алгоритма. Реальные задачи подтверждают практическую полезность предложенного метода.

### 4.1. Эксперименты на искусственных задачах

Для проведения экспериментов была сконструирована масштабируемая модельная задача *Mod-n*. На основе этой задачи исследовались зависимости времени решения и количества итераций от размерности задачи, представительности набора образцов, радиуса и мощности рандомизации. Эксперименты проводились на персональном компьютере под управлением Windows XP со следующими характеристиками: процессор – Intel Pentium IV 3 МГц, оперативная память – 2 Мб, компилятор – Borland C++ version 5.02.

#### 4.1.1. Модельная задача *Mod-n*

Модельная задача *Mod-n* имеет вид, изображенный на Рис. 17. Здесь  $Q_{\max}$  обозначает целевую функцию, для которой необходимо найти максимальное значение. Функция  $\varphi(x)$  моделирует эксперта в том смысле, что она явно задает неформализованное ограничение  $\varphi(x) \leq 0$ . Параметр  $n$  задает размерность пространства  $\mathbb{R}^n$ , в котором решается задача. Для всех  $n \geq 2$  задача *Mod-n* имеет следующее единственное точное решение  $\bar{x}$ :  $\bar{x}_1 = 720$ ,  $\bar{x}_2 = \dots = \bar{x}_n = 640$ .

$$\begin{cases}
 x_1 - 2x_2 \leq 0 \\
 x_1 - 2x_3 \leq 0 \\
 \vdots \\
 x_1 - 2x_n \leq 0 \\
 x_1 + 2x_2 \leq 2000 \\
 x_1 + 2x_3 \leq 2000 \\
 \vdots \\
 x_1 + 2x_n \leq 2000 \\
 x_1 \geq 0, x_2 \geq 0, \dots, x_n \geq 0
 \end{cases}$$

$$Q_{\max} = x_1$$

$$\varphi(x) = 2(n-1)x_1 - x_2 - x_3 - \dots - x_n - 800(n-1)$$

**Рис. 17.** Модельная задача *Mod-n*.

В качестве начального набора образцов фигурирует множество всех вершин выпуклого многогранника, задаваемого формализованными ограничениями задачи *Mod-n*:

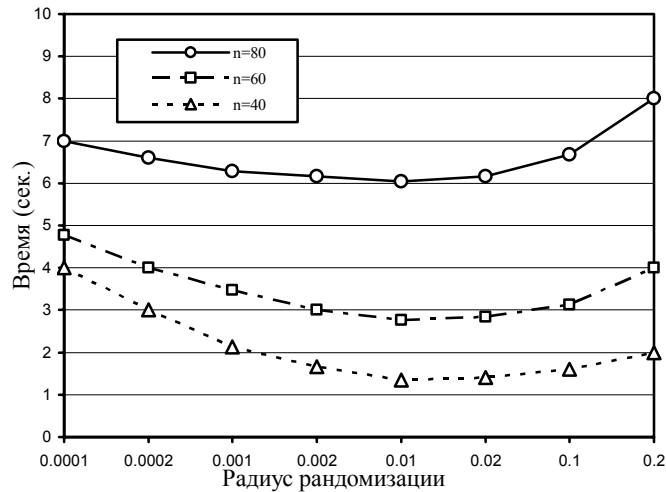
$$\mathbb{A} = \{(1000, 500, \dots, 500)\};$$

$$\mathbb{B} = \{(0, \dots, 0); (0, 1000, 0, \dots, 0); \dots; (0, \dots, 0, 1000); (0, 1000, \dots, 1000)\}.$$

При этом  $|\mathbb{A}| = 1$ ,  $|\mathbb{B}| = n + 1$ .

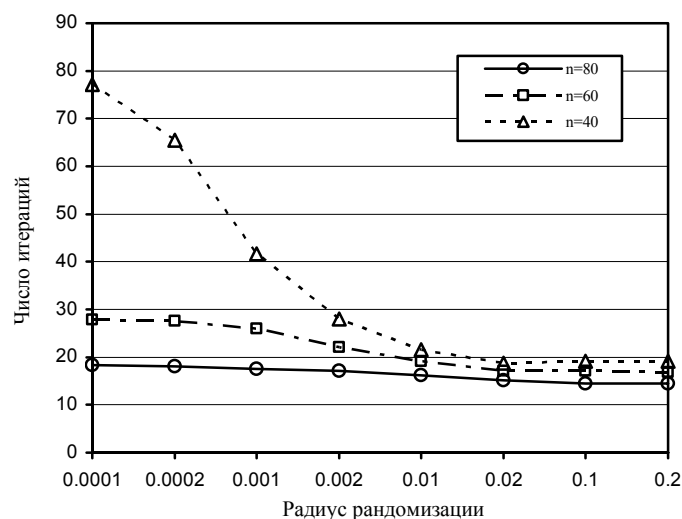
#### 4.1.2. Влияние радиуса рандомизации на эффективность

В первой серии экспериментов была исследована зависимость количества итераций и времени решения задачи *Mod-n* от радиуса рандомизации  $\rho$ . Этот параметр был введен в разделе 2.4. Задача *Mod-n* решалась для значений  $n$ , равных 40, 60 и 80. Результаты проведенных экспериментов представлены на Рис. 18 и Рис. 19. Графики на Рис. 18 показывают, что время решения задачи слабо зависит от радиуса рандомизации, достигая минимума приблизительно при значении  $\rho = 0.01$ .



**Рис. 18.** Зависимость времени решения задачи от радиуса рандомизации  $\rho$ .

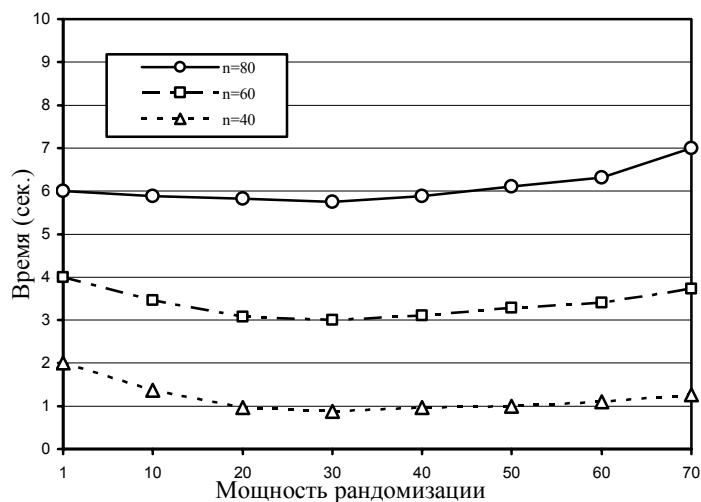
Графики на Рис. 19 показывают, что количество итераций для задач небольшой размерности ( $n = 40$ ) может значительно уменьшаться с увеличением радиуса рандомизации, принимая горизонтально-асимптотический характер для значений  $\rho \geq 0.01$ . Для задач большой размерности ( $n = 80$ ) влияние радиуса рандомизации на количество итераций становится незначительным. Проведенные эксперименты показывают, что в большинстве случаев хорошим выбором является значение  $\rho = 0.01$ .



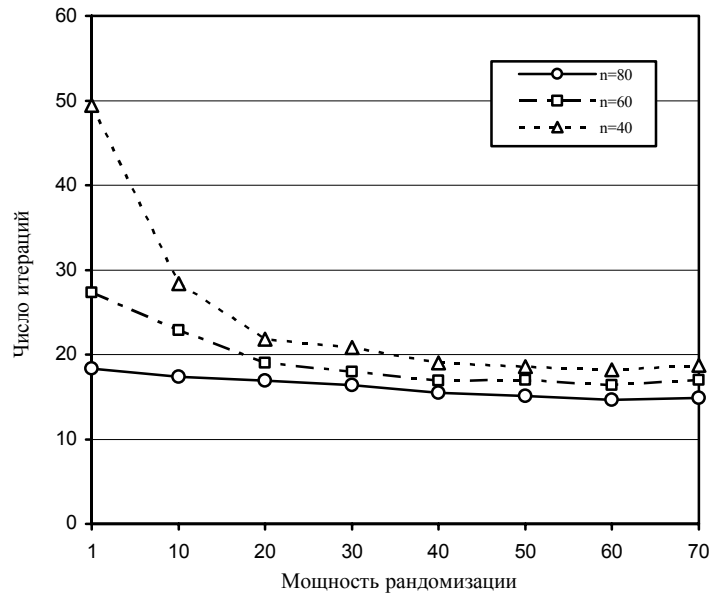
**Рис. 19.** Зависимость числа итераций от радиуса рандомизации  $\rho$ .

### 4.1.3. Влияние мощности рандомизации на эффективность

Во второй серии экспериментов исследовалась зависимость количества итераций и времени решения задачи *Mod-n* от мощности рандомизации  $m$ . Данный параметр был введен в разделе 2.4 и обозначает количество точек, генерируемых процедурой рандомизации. Задача *Mod-n* решалась для значений  $n$ , также равных 40, 60 и 80. Результаты этих экспериментов представлены на Рис. 20 и Рис. 21. Графики на Рис. 20 показывают, что время решения задачи также слабо зависит от мощности рандомизации, достигая минимума приблизительно при значении  $m = 30$ . Графики на Рис. 21 показывают, что количество итераций для всех трех размерностей уменьшается с ростом  $m$ , принимая горизонтально-асимптотический характер для значений  $m > 20$ . При этом для задач большой размерности ( $n = 80$ ) влияние мощности рандомизации на количество итераций также становится незначительным. Проведенные эксперименты показывают, что в большинстве случаев хорошим выбором является значение  $m = 30$ .



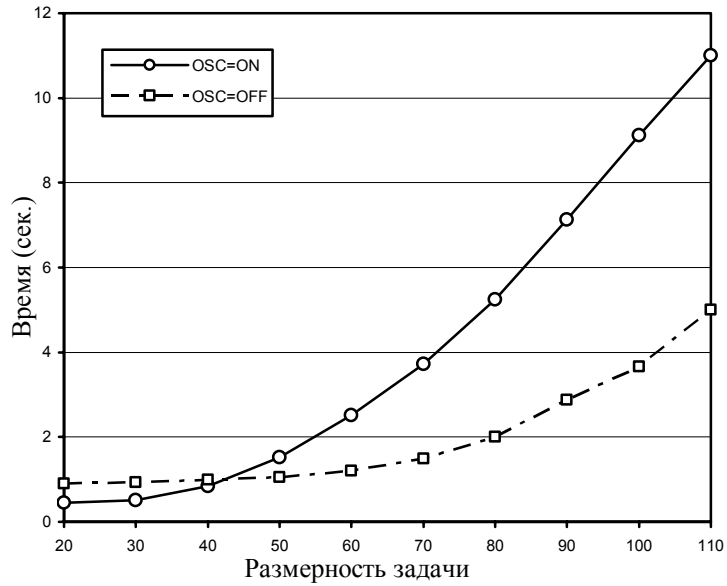
**Рис. 20.** Зависимость времени решения задачи от мощности рандомизации  $m$ .



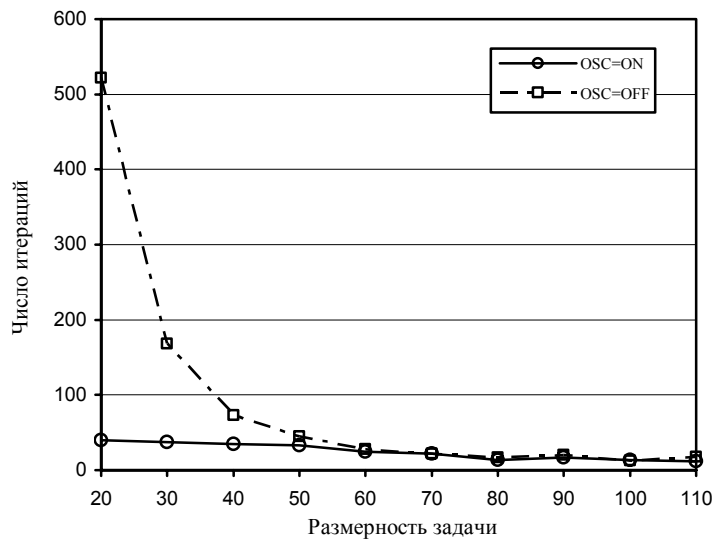
**Рис. 21.** Зависимость числа итераций от мощности рандомизации  $m$ .

#### 4.1.4. Эффективность метода осцилляций

В следующей серии экспериментов исследовалась эффективность применения метода осцилляций при решении задач различной размерности. Были проведены вычислительные эксперименты с задачей *Mod-n*, в ходе которых размерность задачи  $n$  варьировалась в диапазоне от 20 до 110. При этом использовались два варианта алгоритма ЛП-ДА. В первом варианте метод осцилляций был *включен* (OSC=ON), а во втором – *выключен* (OSC=OFF). Результаты этих экспериментов показывают (см. Рис. 22 и Рис. 23), что в обоих вариантах при увеличении размерности время решения задачи растет, а количество итераций уменьшается. Однако, если задача имеет размерность  $n < 50$ , вариант алгоритма, использующий метод осцилляций, позволяет получить решение при значительно меньшем количестве итераций, что дает возможность существенно сократить число обращений к эксперту.



**Рис. 22.** Зависимость времени решения задачи от размерности  $n$ .



**Рис. 23.** Зависимость числа итераций от размерности  $n$ .

#### 4.1.5. Эксперименты с неполными наборами образцов

В данной серии экспериментов исследовалось влияние мощности начального набора образцов на эффективность метода ЛП-ДА. В разделе 2.2 был предложен метод, в соответствии с которым в качестве начального на-

бора образцов берутся все вершины многогранника  $D$ , ограничивающего допустимое множество точек формализованной части исходной задачи. На практике не всегда возможно вычислить все вершины многогранника  $D$ . В связи с этим является важным вопрос, на сколько метод ЛП-ДА является чувствительным к неполным наборам образцов.

Был проведен следующий эксперимент на модельной задаче *Mod-50*. Пусть  $\mathfrak{D}$  обозначает множество всех вершин многогранника  $D$ . Определим представительность начального набора образцов  $\mathfrak{P}$  следующим образом:

$$\text{rep}(\mathfrak{P}) = \frac{|\mathfrak{P}|}{|\mathfrak{D}|}.$$

В частности,  $\text{rep}(\mathfrak{P}) = 1$  при  $\mathfrak{P} = \mathfrak{D}$ , и  $\text{rep}(\mathfrak{P}) = 0$  при  $\mathfrak{P} = \emptyset$ . При проведении эксперимента значение  $\text{rep}(\mathfrak{P})$  варьировалось в диапазоне от 1.0 до 0.6 с шагом 0.5. В каждом случае оценивалась величина  $\Delta = \|\bar{x} - \tilde{x}\|$ , задающая расстояние от точного решения  $\bar{x}$  до приближенного решения  $\tilde{x}$ , полученного в ходе вычислительного эксперимента. Нужные значения  $\text{rep}(\mathfrak{P})$  получались путем удаления соответствующего количества точек из множества  $\mathfrak{B}$ , определенного в разделе 4.1.1. Было проведено две серии испытаний. В первой использовался режим с включенной осцилляцией (OSC=ON), во второй – с отключенной (OSC=OFF).

**Табл. 5.** Влияние начального набора образцов на точность решения.

$\text{rep}(\mathfrak{P})$	$\Delta$	
	OSC=ON	OSC=OFF
1.0	0.0001	0.021
0.95	0.001	3.5
0.9	0.01	6.2
0.85	0.034	9.76
0.8	0.023	12.4
0.75	0.04	16.67
0.7	0.045	19.55
0.65	0.04	22.8
0.6	0.04	27.3

Результаты испытаний сведены в Табл. 5. Полученные экспериментальные данные показывают, что метод осцилляций в значительной мере повышает устойчивость алгоритма ЛП-ДА к неполным начальным наборам образцов.

#### 4.2. Эксперименты на реальной задаче

Эффективность алгоритма ЛП-ДА была также проверена на реальной задаче управления металлургическим предприятием, заимствованной из монографии [44]. Назовем ее задача *Metal*.

Задача *Metal* моделирует управление прокатным комплексом, включающим рельсобалочный стан, участок отделки фасонных профилей и термомоямы. Номенклатура производимой продукции, удельные затраты времени работы оборудования (час/тыс. т) и прибыль (руб/т), получаемая от реализации продукции, приводятся в Табл. 6.

Табл. 6. Информационное описание прокатного комплекса.

Оборудование	Продукция				
	Осевые заготовки	Рельсы	Балки	Конструкционный прокат	Квадратные заготовки
Рельсобалочный стан	6.4	4.0	7.5	7.0	4.4
Участок отделки	0	0	35	20	0
Термомоямы	85	0	0	75	18
Прибыль	18	16.8	26.3	36.2	17.5

Фонд фактического времени работы оборудования в рассматриваемый период для всех агрегатов и участков составляет 7 тыс. час. Согласно проекту планового задания, необходимо произвести как максимум (в тыс. т): осевых заготовок – 50, рельсов – 1150, балок – 150, конструкционного проката – 5. Производство квадратных заготовок фиксировано и составляет 125 тыс. т.

Необходимо максимизировать эффективность функционирования рельсо-балочного комплекса.

Сформулируем задачу ЛПНО для данного примера. Обозначим через  $x_i$  объем  $i$ -го вида производства. По проекту планового задания для них заданы ограничения

$$x_1 \leq 50, \quad x_2 \leq 1150, \quad x_3 \leq 150, \quad x_4 \leq 5, \quad x_5 = 125.$$

С учетом Табл. 6 формализованная часть нашей задачи описывается следующей системой неравенств:

$$\begin{cases} 6.4x_1 + 4x_2 + 7.5x_3 + 7x_4 + 4.4x_5 \leq 7000 \\ 35x_3 + 20x_4 \leq 7000 \\ x_1 \leq 50 \\ x_2 \leq 1150 \\ x_3 \leq 150 \\ x_4 \leq 5 \\ x_5 = 125 \end{cases}$$

В качестве неформализованного ограничения фигурирует неравенство

$$85x_1 + 75x_4 \leq y.$$

Здесь  $y$  – константа, определяемая экспертом по неформализуемым критериям. Поскольку необходимо максимизировать прибыль, то целевую функцию представим в виде

$$Q_{\max} = 18x_1 + 16.8x_2 + 26.3x_3 + 36.2x_4 + 17.5x_5.$$

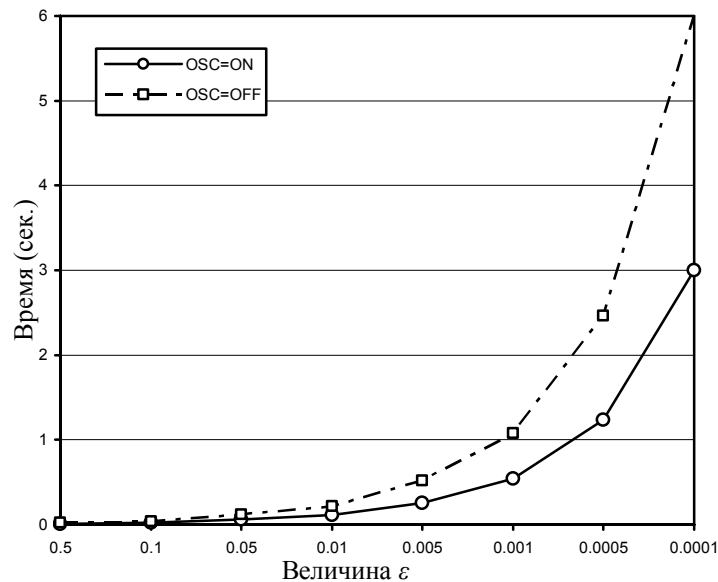
Множества  $\mathbb{A}$  и  $\mathbb{B}$  задаются как множества недопустимых и допустимых по неформализованному ограничению планов. Для  $y = 4750$  в качестве набора образцов были взяты следующие множества точек:

$$\begin{aligned} \mathbb{A} = \{ & (50, 1150, 150, 58); \\ & (108, 1150, 150, 5); \\ & (53, 1150, 197, 5); \\ & (50, 1150, 194, 11) \}; \end{aligned}$$

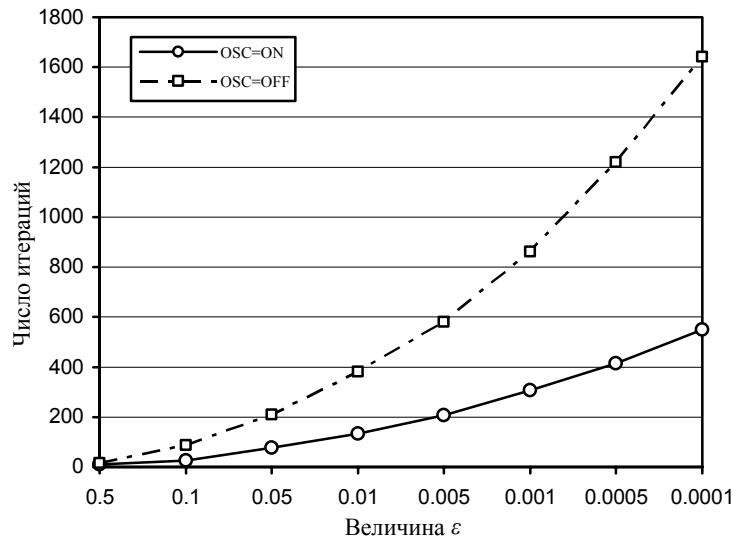
$$\mathbb{B} = \left\{ \begin{array}{l} (50, 1243, 150, 5); \\ (50, 1154, 197, 5); \\ (0, 0, 0, 0) \end{array} \right\}.$$

Данные точки являются вершинами многогранника, ограничивающего область допустимых значений формализованной части задачи *Metal*. Они были получены путем решения задачи ЛПФ симплекс-методом для различных значений коэффициентов целевой функции.

С задачей *Metal* была проведена серия вычислительных экспериментов, в которых исследовались зависимость времени решения задачи и количества итераций от значения  $\varepsilon$ , используемого в критерии завершения (см. раздел 2.3). Проведенные эксперименты показали (см. Рис. 24 и Рис. 25), что метод осцилляций во всех случаях обеспечивает примерно двукратное превосходство по скорости сходимости.



**Рис. 24.** Зависимость времени решения задачи от величины  $\varepsilon$ .



**Рис. 25.** Зависимость числа итераций от величины  $\varepsilon$ .

### 4.3. Масштабируемость параллельного алгоритма

Важным свойством любого параллельного алгоритма является его масштабируемость [45]. *Масштабируемость* можно определить как меру эффективности распараллеливания алгоритма на многопроцессорных конфигурациях с различным количеством процессорных узлов.

Существуют две основные качественные характеристики эффективности распараллеливания: *ускорение* [74] и *расширяемость* [58]. Дадим следующее формализованное определение указанных понятий.

Пусть даны две различные конфигурации  $A$  и  $B$  многопроцессорной системы с заданной архитектурой [4], различающиеся количеством процессоров и ассоциированных с ними устройств (мы подразумеваем, что все конфигурации предполагают пропорциональное наращивание модулей памяти и дисков). Пусть задана некоторая задача  $Q$ . *Коэффициент ускорения*  $a_{AB}$ , получаемый при переходе от конфигурации  $A$  к конфигурации  $B$ , определяется следующей формулой

$$a_{AB} = \frac{d_{At_{QA}}}{d_{Bt_{QB}}},$$

где  $d_A$  – степень параллелизма (количество процессоров) конфигурации  $A$ ;  $d_B$  – степень параллелизма конфигурации  $B$ ;  $t_{QA}$  – время, затраченное конфигурацией  $A$  на выполнение задачи  $Q$ ;  $t_{QB}$  – время, затраченное конфигурацией  $B$  на выполнение задачи  $Q$ .

Говорят, что задача  $Q$  демонстрирует *линейное ускорение*, если коэффициент ускорения остается равным единице для всех конфигураций с данной архитектурой.

Пусть теперь задан набор однопоточных задач  $Q_1, Q_2, \dots$ , количественно превосходящих некоторую фиксированную задачу  $Q$  в  $i$  раз, где  $i$  – номер соответствующей задачи. Пусть заданы конфигурации многопроцессорной системы с заданной архитектурой  $A_1, A_2, \dots$ , превосходящие по степени параллелизма некоторую минимальную конфигурацию  $A$  в  $j$  раз, где  $j$  – номер соответствующей конфигурации. Тогда коэффициент расширяемости  $e_{km}$ , получаемый при переходе от конфигурации  $A_k$  к конфигурации  $A_m$  ( $k < m$ ), задается формулой

$$e_{km} = \frac{t_{Q_k A_k}}{t_{Q_m A_m}},$$

где  $t_{Q_k A_k}$  – время, затраченное конфигурацией  $A_k$  на выполнение задачи  $Q_k$ ;  $t_{Q_m A_m}$  – время, затраченное конфигурацией  $A_m$  на выполнение задачи  $Q_m$ . Говорят, что многопроцессорная система демонстрирует *линейную расширяемость* на наборе задач  $Q_1, Q_2, \dots$ , если коэффициент расширяемости остается равным единице для всех конфигураций данной системы.

Ускорение позволяет определить эффективность наращивания системы на сопоставимых задачах. Расширяемость позволяет измерить эффективность наращивания системы на больших задачах.

Говорят, что параллельный алгоритм *хорошо масштабируем*, если он демонстрирует ускорение и расширяемость, близкие к линейным.

Для исследования масштабируемости параллельной версии алгоритма ЛП-ДА была сконструирована следующая модельная задача ЛПНО\*, получившая название  $Mod-n_r$ :

$$\begin{cases} x_1 - 2x_2 \leq 0 \\ x_1 - 2x_3 \leq 0 \\ \vdots \\ x_1 - 2x_n \leq 0 \\ x_1 + 2x_2 \leq 8000 \\ x_1 + 2x_3 \leq 8000 \\ \vdots \\ x_1 + 2x_n \leq 8000 \\ x_1 \geq 0, x_2 \geq 0, \dots, x_n \geq 0 \end{cases}$$

$$Q_{\max} = x_1$$

Неформализованные ограничения задачи  $Mod-n_r$  задавались следующими экспертными функциями:

$$\begin{cases} \mathbf{e}_1(x) = \text{sgn}(3x_1 - x_2 - 3000) \\ \vdots \\ \mathbf{e}_r(x) = \text{sgn}(3x_1 - x_r - 3000) \end{cases}$$

Предполагается, что  $r \leq n-1$ .

В качестве начального набора образцов  $\mathfrak{P} = \{\mathbb{A}, \mathbb{B}\}$  были взяты множества точек

$$\begin{aligned} \mathbb{A} = \{ & (2500, 1250, 1250, \dots, 1250); \\ & (2500, 2750, 1250, \dots, 1250); \\ & (2500, 1250, 2750, \dots, 1250); \\ & \vdots \\ & (2500, 1250, 1250, \dots, 2750); \\ & (2500, 2750, 2750, \dots, 2750) \} \end{aligned}$$

и

$$\mathbb{B} = \{ (1000, 500, 500, \dots, 500); \\ (1000, 3500, 500, \dots, 500); \\ (1000, 500, 3500, \dots, 500); \\ \vdots \\ (1000, 500, 500, \dots, 3500); \\ (1000, 3500, 3500, \dots, 3500) \}$$

Имеем  $|\mathbb{A}| = n + 1$  и  $|\mathbb{B}| = n + 1$ . Таким образом, задача  $Mod-n_r$  представляет собой задачу ЛПНО\* в пространстве  $\mathbb{R}^n$ , имеющую  $r$  неформализованных ограничений.

Для проведения экспериментов был использован вычислительный кластер [35] с характеристиками, представленными в Табл. 7.

**Табл. 7.** Характеристики вычислительного кластера.

Количество вычислительных узлов	26	
Количество процессоров	52	
Конфигурация узла	Процессоры	2 × Intel Xeon64 DP 3,2 GHz
	RAM	2 Gb
	HDD	80 Gb
Тип системной сети	InfiniBand (PCI-Express 4X)	
Тип управляющей (вспомогательной) сети	Gigabit Ethernet	
Операционная система	SUSE Linux Enterprise Server 9.1	
MPI-2	MVAPICH2	
Компилятор	Intel C/C++ Compiler 9.0 EMT 64	

В рамках диссертационной работы на указанном вычислительном кластере были исследованы ускорение и расширяемость, получаемые при использовании параллельного алгоритма ЛП-ДА на базе симплекс-метода и метода линейной коррекции. Результаты по исследованию ускорения представлены на Рис. 26. Эксперименты проводились для трех размерностей:  $n = 60$ ,  $n = 80$  и  $n = 100$  при фиксированном количестве неформализованных ограничений  $r = 48$ . Ускорение вычислялось по формуле

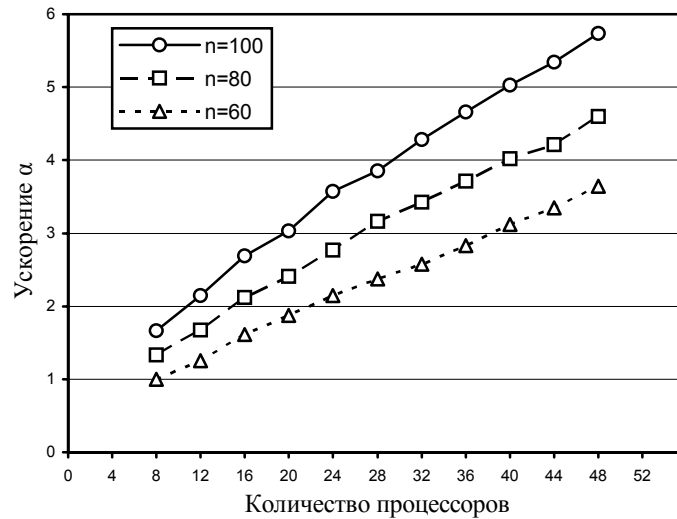


Рис. 26. Ускорение для  $Mod-n_{48}$ .

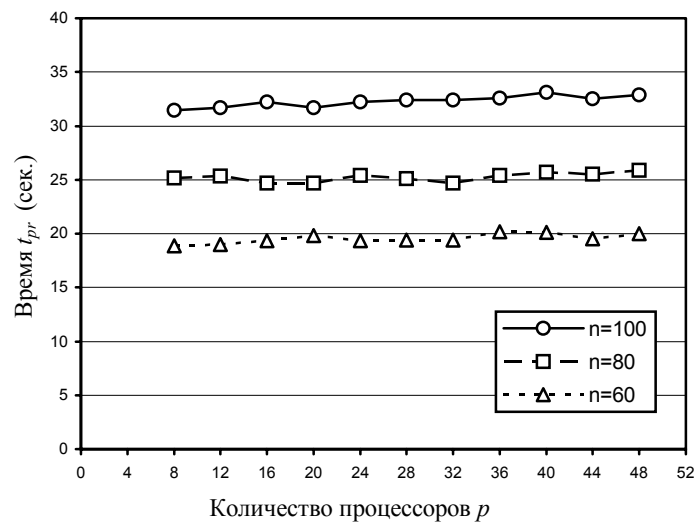


Рис. 27. Расширяемость для  $Mod-n_p$ .

$\alpha = \alpha_0 + t_p/t_8$ , где  $t_8$  – время, затраченное на решение задачи  $Mod-n_{48}$  на 8 процессорах,  $t_p$  – время, затраченное на решение этой же задачи на  $p$  процессорах,  $\alpha_0 = (n - 60)/60$ . Для каждой размерности варьировалось количество процессоров, используемых для решения задачи. Во всех трех случаях было получено ускорение, близкое к линейному.

Результаты по исследованию расширяемости для параллельного алгоритма ЛП-ДА приведены на Рис. 27. Эксперименты проводились также

для трех размерностей:  $n = 60$ ,  $n = 80$  и  $n = 100$ . Здесь  $t_{pr}$  обозначает время, затраченное на решение модельной задачи размерности  $n$  с  $r$  неформализованными ограничениями на  $p$  процессорах. При проведении эксперимента для каждой размерности варьировалось количество процессоров, используемых для решения задачи. При этом количество неформализованных ограничений всегда равнялось количеству процессоров. Проведенные эксперименты показали, что во всех случаях параллельный алгоритм ЛП-ДА демонстрировал расширяемость, близкую к линейной.

Таким образом, можно сделать вывод о том, что параллельный алгоритм ЛП-ДА показывает хорошую масштабируемость на многопроцессорных системах с массовым параллелизмом при решении задач ЛПНО\* с различным количеством неформализованных ограничений.

## ЗАКЛЮЧЕНИЕ

В диссертационной работе были рассмотрены вопросы, связанные с решением задач линейного программирования при наличии неформализованных ограничений. Было показано, что задачу с несколькими неформализованными ограничениями можно свести к задаче с одним неформализованным ограничением. Предложен общий метод для решения задач с одним неформализованным ограничением, базирующийся на дискриминантном анализе и экспертизе новых образцов, получаемых в результате решения задачи линейного программирования. Каждый новый образец получается как решение аппроксимационной задачи линейного программирования, где вместо неформализованного ограничения фигурирует неравенство на основе разделяющей функции. В случае, когда решение аппроксимационной задачи уже присутствует в наборе образцов, применяется процедура рандомизации. Для описанного метода была доказана теорема сходимости и рассмотрены вопросы его практической реализации. В качестве реализации общего метода предложен итерационный алгоритм ЛП-ДА для решения задач линейного программирования при наличии нескольких неформализованных ограничений. Этот алгоритм базируется на сочетании симплекс-метода и метода линейной коррекции. Рассмотрены различные аспекты реализации алгоритма ЛП-ДА в виде программы для ЭВМ. Предложена параллельная версия алгоритма. Выполнены проектирование и реализация программного комплекса на языке Си и проведено большое количество вычислительных экспериментов на реальных и искусственных задачах, подтверждающих эффективность предложенных подходов.

Работа выполнялась при поддержке *Российского фонда фундаментальных исследований* (проекты 03-01-00565, 06-01-00380).

В заключение перечислим основные полученные результаты диссертационной работы, приведем данные о публикациях и апробациях, и рассмотрим направления дальнейших исследований в данной области.

## **ОСНОВНЫЕ РЕЗУЛЬТАТЫ ДИССЕРТАЦИОННОЙ РАБОТЫ**

На защиту выносятся следующие новые научные результаты.

1. Предложен и исследован итерационный метод для решения задач линейного программирования с неформализованными ограничениями, базирующийся на синтезе дискриминантного анализа и линейной оптимизации.
2. Предложен метод осцилляций, позволяющий существенно ускорить сходимость последовательности приближений к точному решению.
3. Разработан и исследован алгоритм ЛП-ДА, соединяющий методы линейного программирования и дискриминантного анализа.
4. Спроектирован и реализован прототип программного комплекса для решения задач линейного программирования с неформализованными ограничениями. Общий объем кода на языке Си составил около 3000 строк. Исходные тексты прототипа свободно доступны в Интернет по адресу: <http://life.susu.ru/lpno/>.
5. Спроектирована и реализована параллельная версия алгоритма ЛП-ДА с использованием пакета MPI-2. Исходные тексты параллельной версии свободно доступны в Интернет по адресу: <http://life.susu.ru/lpnoMPI/>.
6. Проведены вычислительные эксперименты на модельных и реальных задачах экономико-математического моделирования, подтверждающие эффективность предложенных алгоритмов, методов и подходов.

## ПУБЛИКАЦИИ ПО ТЕМЕ ДИССЕРТАЦИИ

Основные результаты диссертации полностью опубликованы в следующих печатных работах.

1. *Соколинская И.М.* Синтез симплекс-метода и метода линейной коррекции в задачах линейной оптимизации с неформализованными ограничениями // Вычислительные методы и программирование. –2005. –Том 6, №2. –С. 103-115.
2. *Mazurov V.I., Sokolinskaya I.M.* Discrimination analysis and randomization in linear optimization problems with not formalized restrictions // Pattern Recognition and Image Analysis. –2006. –Vol. 16, No. 2. –P. 170-178.
3. *Соколинская И.М.* Метод осцилляций в задачах линейного программирования с неформализованным ограничением // Алгоритмический анализ неустойчивых задач: Тез. докл. Всерос. науч. конф. (2-6 февраля 2004 г., Екатеринбург). -Екатеринбург: Изд.-во Урал. ун-та. -2004. -С. 302-303.
4. *Соколинская И.М., Соколинский Л.Б.* Программный комплекс для решения задач линейного программирования с неформализованными ограничениями // Первая Международная Конференция «Системный Анализ и Информационные Технологии» САИТ-2005 (12-16 сентября 2005 г., Переславль-Залесский, Россия): Труды конференции. В 2 т. Т. 2. –М.: КомКнига. -2005. -С. 286-292.
5. *Соколинская И.М., Соколинский Л.Б.* Параллельный алгоритм решения задачи линейного программирования с неформализованными ограничениями // III Всероссийская конференция "Проблемы оптимизации и экономические приложения": Материалы конференции (Омск, 11-15 июля 2006) / Омский филиал Института математики им. С.Л. Соболева СО РАН. -Омск: Изд-во ОмГТУ. -2006. С. 156.

## АПРОБАЦИЯ РАБОТЫ

Основные положения диссертационной работы, разработанные модели, методы, алгоритмы и результаты вычислительных экспериментов докладывались автором на следующих международных и всероссийских научных конференциях:

- на Всероссийской научной конференции "Алгоритмический анализ неустойчивых задач" (2-6 февраля 2004 г., Екатеринбург);
- на Международной научной конференции "Системный Анализ и Информационные Технологии" САИТ-2005 (12-16 сентября 2005 г., Переславль-Залесский);
- на III Всероссийской научной конференции "Проблемы оптимизации и экономические приложения" (Омск, 11-15 июля 2006 г.).

## НАПРАВЛЕНИЯ ДАЛЬНЕЙШИХ ИССЛЕДОВАНИЙ

Теоретические исследования и практические разработки, выполненные в рамках этой диссертационной работы, предполагается продолжить по следующим направлениям.

1. Доказательство сходимости представленного в данной диссертации алгоритма  $\mathcal{L}$  при более слабых ограничениях. В частности, может быть сформулирована гипотеза о том, что теорема сходимости для алгоритма  $\mathcal{L}$  остается справедливой и в случае устранения условия нормализации. В пользу этой гипотезы свидетельствуют теорема 3 (см. раздел 1.6 диссертации) и проведенные в ходе диссертационного исследования эксперименты на реальных и искусственных задачах (см. главу 4 диссертации), так как во всех экспериментах был использован вариант алгоритма без условия нормализации.
2. Создание версии программного комплекса для 64-разрядных процессоров. Здесь основной проблемой является подбор оптимальных значений допусков, связанных с преодолением ошибок округления.

3. Создание в рамках программного комплекса альтернативных блоков для решения задач линейного программирования и дискриминантного анализа (в дополнение к симплекс-методу и методу линейной коррекции предполагается реализовать методы фейеровского типа [2, 3, 9, 10] и метод комитетов [20, 26, 27, 46]).

## ЛИТЕРАТУРА

1. *Бердникова Е.А.* Параллельный алгоритм решения задачи линейного программирования на основе оператора проектирования на линейное многообразие // Высокопроизводительные вычисления и их приложения: Труды Всероссийск. науч. конф. (30 октября - 2 ноября 2000 г., г. Черноголовка). -М.: Изд-во МГУ. -2000. -С. 71-72.
2. *Бердникова Е.А., Ерёмин И.И., Попов Л.Д.* Распределенные фейеровские процессы для систем линейных неравенств и задач линейного программирования // Автоматика и телемеханика. -№. 2. -2004. -С. 16-32.
3. *Васин В.В., Ерёмин И.И.* Операторы и итерационные процессы фейеровского типа. Теория и приложения. –Екатеринбург: УрО РАН, 2005. -210 с.
4. *Воеводин Вл.В., Капитонова А.П.* Методы описания и классификации архитектур вычислительных систем. —М: Изд-во МГУ, 1994. -103 с.
5. *Воеводин В.В., Воеводин Вл.В.* Параллельные вычисления. -СПб.: БХВ-Петербург, 2002. -600 с.
6. Горелик А.Л., Гуревич И. Б., Скрипкин В. А. Современное состояние проблемы распознавания: Некоторые аспекты. -М.: Радио и связь, 1985. -161 с.
7. *Данциг Дж.* Линейное программирование, его применение и обобщения. -М.: Прогресс, 1966. -600 с.
8. *Джесксон П.* Введение в экспертные системы. -М: Издательский дом "Вильямс", 2001. -624 с.
9. *Ерёмин И.И.* Методы фейеровских приближений в выпуклом программировании // Мат. заметки. -1968. -Т. 3, вып. 2. -С. 217-234.

10. *Еремин И.И.* Применение метода фейеровских приближений к решению задач выпуклого программирования с негладкими ограничениями // Журн. вычисл. мат. и мат. физики. -1969. -Т. 9, № 5. -С. 1153-1160.
11. *Еремин И.И.* Общая теория устойчивости в линейном программировании // Известия ВУЗов. Математика. -1999. -N 12. -С. 43-52.
12. *Еремин И.И.* Теория линейной оптимизации. –Екатеринбург: Изд-во “Екатеринбург”, 1999. -312 с.
13. *Еремин И.И.* Фейеровские методы сильной отделимости выпуклых полиэдральных множеств // Известия вузов. Сер. математика. -2006. (в печати).
14. *Еремин И.И., Астафьев Н.Н.* Введение в теорию линейного и выпуклого программирования. -М.: Наука, 1976. -192 с.
15. *Еремин И.И., Мазуров Вл.Д.* Нестационарные процессы математического программирования. –М.: Наука, 1979. -291 с.
16. *Еремин И. И., Мазуров Вл. Д.* Вопросы оптимизации и распознавания образов. -Свердловск: Сред.-Урал. кн. изд-во, 1979. -63 с.
17. *Еремин И.И., Мазуров Вл.Д., Скарин В.Д., Хачай М.Ю.* Математические методы в экономике. -Екатеринбург: У-Фактория, 2000. -280 с.
18. *Канторович Л.В.* Математические методы организации и планирования производства. –Л.: Изд-во ЛГУ, 1939. -68 с.
19. *Корнеев В.В.* Параллельные вычислительные системы. -М.: “Нолидж”, 1999. -320 с.
20. *Мазуров Вл. Д.* Комитеты систем неравенств и задача распознавания // Кибернетика. -1971. -№ 3. -С. 140-146.
21. *Мазуров Вл. Д.* Об одном итерационном методе планирования, использующем распознавание образов для учета плохо формализуемых факторов // Изв. АН СССР. Техн. кибернетика. -1973. - № 3. -С. 205-207.

22. *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.
23. *Мазуров Вл.Д.* Дискриминантный анализ при математическом моделировании плохо формализуемых ситуаций // Нелинейная оптимизация и приложения в планировании. –Свердловск: УНЦ АН СССР, 1973. -С. 26-35.
24. *Мазуров Вл.Д.* Комитеты в нечетких задачах // Методы оптимизации и распознавания образов в задачах планирования. -Свердловск: УНЦ АН СССР, 1980. -С. 44-65.
25. *Мазуров Вл.Д.* О задаче оптимизации с плохо формализуемой целью // Параметрическая оптимизация. –Свердловск: УНЦ АН СССР, 1985. -С. 51-53
26. *Мазуров Вл.Д.* Метод комитетов в задачах оптимизации и классификации. -М.: Наука, 1990. -248 с.
27. *Мазуров Вл.Д., Кривоногов А.И., Казанцев В.С., Сачков Н.О., Белецкий Н.Г.* Комитеты в принятии решений // Кибернетика. -1984. -№ 1. -С. 90-95.
28. *Мазуров Вл.Д., Мазуров П.В.* Оптимизация, распознавание и нейронные сети в экономике. –Екатеринбург: УрГУ, 1999. -58 с.
29. *Мак-Каллок У.С., Питтс В.* Логическое исчисление идей, относящихся к нервной деятельности // Нейрокомпьютер. –1992. –№ 3–4. –С. 29–34.
30. *Меламед И.И.* Нейронные сети и комбинаторная оптимизация // Автоматика и телемеханика. – 1994. – № 11. – С. 3–40.
31. *Муртаф Б.* Современное линейное программирование: Теория и практика. -М.: Мир, 1984. -224 с.

32. *Немчинов В.С.* Экономико-математические методы и модели. –Т. 3. -М.: Наука, 1967.
33. *Нильсон Н.* Обучающиеся машины. –М.: Мир, 1967. -180 с.
34. *Оленев Н.Н.* Основы параллельного программирования в системе MPI. -М.: изд-во ВЦ РАН, 2005. 90 с.
35. Портал вычислительного кластера «Infinity» [<http://cluster.susu.ru/>].
36. *Поспелов Д.А.* Логико-лингвистические модели в системах управления. -М.: Энергоиздат, 1981. -232 с.
37. *Розин Б.Б.* Теория распознавания образов в экономических исследованиях. –М.: Статистика, 1973.
38. *Соколинская И.М.* Синтез симплекс-метода и метода линейной коррекции в задачах линейной оптимизации с неформализованными ограничениями // Вычислительные методы и программирование. -2005. -Том 6, №2. -С. 103-115.
39. *Соколинская И.М.* Метод осцилляций в задачах линейного программирования с неформализованным ограничением // Алгоритмический анализ неустойчивых задач: Тез. докл. Всерос. науч. конф. (2-6 февраля 2004 г., Екатеринбург). -Екатеринбург: Изд.-во Урал. ун-та. -2004. -С. 302-303.
40. *Соколинская И.М., Соколинский Л.Б.* Параллельный алгоритм решения задачи линейного программирования с неформализованными ограничениями // III Всероссийская конференция "Проблемы оптимизации и экономические приложения": Материалы конференции (Омск, 11-15 июля 2006) / Омский филиал Института математики им. С.Л. Соболева СО РАН. -Омск: Изд-во ОмГТУ. -2006. -С. 156.

41. *Соколинская И.М., Соколинский Л.Б.* Программный комплекс для решения задач линейного программирования с неформализованными ограничениями // Первая Международная Конференция «Системный Анализ и Информационные Технологии» САИТ-2005 (12-16 сентября 2005 г., Переславль-Залесский, Россия): Труды конференции. В 2 т. Т. 2. –М.: КомКнига. -2005. -С. 286-292.
42. *Соколинский Л.Б., Цымблер Н.Ю.* Параллельный алгоритм решения задач линейного программирования на основе фейеровских отображений. Технический отчет РФФИ#06-01-00380/У1Н1. –Челябинск: ЮУрГУ, 2006.
43. *Таненбаум Э.* Современные операционные системы. -Издательство: Питер, 2002. -1040 с.
44. *Фролов В.Н.* Оптимизация плановых программ при слабо согласованных ограничениях. –М.: Наука, 1986. -164 с.
45. *Четверушкин Б.Н.* Высокопроизводительные многопроцессорные вычислительные системы // Вестник российской академии наук. -2002. -Том 72, №9. -С. 786-794.
46. *Ablow C.M., Kaylor D.J.* A committee solution of the pattern recognition problem // IEEE Trans. -1965. –V. 71, No. 5.
47. *Bartels R.H., Golub G.H.* The simplex method of linear programming using LU decomposition // Communications of the ACM. -1969. Vol. 12, No. 5. – P. 266-268.
48. *Bishop C.M.* Neural Networks for Pattern Recognition. -Oxford University Press, 1996. -504.
49. *Dantzig G.B., Wolfe P.* Decomposition principle for linear programs // Oper Res.. -1960. -Vol. 8, No. 1. -P. 101-111.

50. *Dongarra J.J., Otto S.W., Snir M., Walker D.* A message passing standard for MPP and workstations // Communications of the ACM. -1996. -Vol. 39, No. 7. -P. 84-90.
51. *Eremin I.I., Mazurov V.I.* Nonstationary Processes for Mathematical Programming Problems under the Conditions of Poorly Formalized Constraints and Incomplete Defining Information // Optimization Techniques, IFIP Technical Conference, Novosibirsk, USSR, July 1-7, 1974. Lecture Notes in Computer Science, Vol. 27. -Springer, 1974. -P. 37-41.
52. *Gass S.I.* Linear Programming. - New York: McGraw-Hill. -1969.
53. *Ghosh J., Hwang K.* Critical issues in mapping neural networks on message-passing multicomputers // Proceedings of the 15th Annual International Symposium on Computer architecture. Honolulu, Hawaii, United States. - 1988. -P. 3-11.
54. *Giarratano J.C., Riley G.D.* Expert Systems: Principles and Programming. -Course Technology. -1998. -624 p.
55. *Gose T., Johnsonbaugh R., Jost S.* Pattern Recognition and Image Analysis. -Prentice Hall, 1996. -483 p.
56. *Gropp W.* MPICH2: A New Start for MPI Implementations // Recent Advances in Parallel Virtual Machine and Message Passing Interface: 9th European PVM/MPI Users' Group Meeting, Linz, Austria, September 29 - October 2, 2002. Proceedings. Lecture Notes in Computer Science, V. 2474. -Springer. -2002.
57. *Gropp W., Huss-Lederman S., Lumsdaine A., Lusk E., Nitzberg B., Saphir W., Snir M.* MPI - The Complete Reference: Volume 2, The MPI Extensions. - MIT Press, 1998.
58. *Gunther N.J.* The Practical Performance Analyst. -Authors Choice Press, 2000. -468 p.

59. *Fausett L.V.* Fundamentals of Neural Networks. -Prentice Hall, 1994. -461 p.
60. *Flynn M.J., Rudd K.W.* Parallel architectures // ACM Computing Surveys. - 1996. -Vol. 28, No. 1. -P. 67-70.
61. *Forrest J.J.H., Tomlin J.A.* Implementing the simplex method for the optimization subroutine library // IBM Systems Journal. -1992. -Vol. 31, No. 1. -P. 11-25.
62. *Hadley G.* Linear Programming. - Mass.: Addison-Wesley, Reading. -1962.
63. *Hopfield J.J., Tank D.W.* Neural computation of decision in optimization problems // Biol. Cybernet. -1985. -V. 52. -P. 141-152.
64. *Jordan M.I., Bishop C.M.* Neural networks // ACM Computing Surveys. -1996. -Vol. 28, No. 1. -P. 73-75.
65. *Kennedy J.V., Austin J., Pack R., Cass B.* CNNAP - A Parallel Processing Architecture for Binary Neural Networks // Proceedings of the IEEE International Conference on Neural Networks (ICNN'95), Perth, Western Australia. -1995.
66. *Lachenbruch P.A.* Discriminant Analysis. -New York: Hafner Press, 1975.
67. *Lyu J.J., Luh H., Lee M.-C.* Solving Linear Programming Problems on the Parallel Virtual Machine Environment // American Journal of Applied Sciences. -2004. -Vol. 1, No. 2. -P. 90-94.
68. *Martinson R.K., Tind J.* An interior point method in Dantzig-Wolfe decomposition // Computers & Operations Research. -1999. -Vol. 26, No. 12. -P. 1195-1216.
69. *Mazurov V.I., Sokolinskaya I.M.* Discrimination analysis and randomization in linear optimization problems with not formalized restrictions // Pattern Recognition and Image Analysis. -2006. -Vol. 16, No. 2. -P. 170-178.

70. *Molina F.W.* A Survey of Resource Directive Decomposition in Mathematical Programming // ACM Computing Surveys. -1979. -Vol. 11, No. 2. -P. 95-104.
71. *McLachlan G.J.* Discriminant Analysis and Statistical Pattern Recognition. -New York: John Wiley and Sons, 1992. -526 p.
72. *Nazareth J.L.* Computer Solution of Linear Programs. -Oxford University Press, 1988.
73. *Orchard-Hays W.* Advanced Linear Programming Computing Techniques. - New York: McGraw-Hill, 1968.
74. *Quinn M.J.* Parallel Computing: Theory and Practice. -McGraw-Hill Companies, 1993. -446 p.
75. *Snir M., Otto S., Huss-Lederman S., Walker D., Dongarra J.* MPI - The Complete Reference. Volume 1, The MPI Core. 2nd Ed. -MIT Press, 1999.
76. *Stunke C.B.I., Reed D.A.* Hypercube implementation of the simplex algorithm // Proceedings of the third conference on Hypercube concurrent computers and applications - Volume 2. -New York, NY, USA: ACM Press. -1989. -P. 1473-1482.
77. *White W.W.* A Status Report on Computing Algorithms for Mathematical Programming // ACM Computing Surveys. -1973. -Vol. 5, No. 3. -P. 135-166.

## ПРИЛОЖЕНИЕ. ОСНОВНЫЕ ОБОЗНАЧЕНИЯ

Обозначение	Значение	Раздел
$\bar{\varphi}(x)$	Аффинная функция, задающая неформализованное ограничение: $\bar{\varphi}(x) \leq 0$	1.2
ЛПНО	Задача линейного программирования с одним неформализованным ограничением: $\max \{(c, x) \mid Ax \leq b, \bar{\varphi}(x) \leq 0\}$	1.1
ЛПНО*	Задача линейного программирования с несколькими неформализованными ограничениями: $\max \{(c, x) \mid Ax \leq b, \bar{\varphi}_1(x) \leq 0, \dots, \bar{\varphi}_q(x) \leq 0\}$	1.7
$\bar{x}$	Решение задачи ЛПНО (ЛПНО*)	1.2 (1.7)
$\epsilon(x)$	Функция эксперта: $\begin{cases} \epsilon(x) = 1 & \Leftrightarrow \bar{\varphi}(x) > 0 \\ \epsilon(x) = 0 & \Leftrightarrow \bar{\varphi}(x) = 0 \\ \epsilon(x) = -1 & \Leftrightarrow \bar{\varphi}(x) < 0 \end{cases}$	1.1
$\mathfrak{P} = \{\mathbb{A}, \mathbb{B}\}$	Набор образцов: $\begin{cases} \epsilon(x) = 1, \forall x \in \mathbb{A} \\ \epsilon(x) = -1, \forall x \in \mathbb{B} \end{cases}$	1.1
$\tilde{\varphi}(x)$	Аффинная функция, разделяющая множества $\mathbb{A}$ и $\mathbb{B}$ : $\tilde{\varphi}(x) > 0$ для $x \in \mathbb{A}$ и $\tilde{\varphi}(x) < 0$ для $x \in \mathbb{B}$	1.2
ЛПА	Аппроксимационная задача линейного программирования: $\max \{(c, x) \mid Ax \leq b, \tilde{\varphi}(x) \leq 0\}$	1.2
$\tilde{x}$	Решение задачи ЛПА	1.2
$D$	Многогранник, определяемый формализованной частью задачи ЛПНО: $D = \{x \mid Ax \leq b\}$	1.2
$\bar{P}$	Полупространство, задающее множество значений, удовлетворяющих неформализованному ограничению: $\bar{P} = \{x \mid \bar{\varphi}(x) \leq 0\}$	1.2
$\bar{H}$	Гиперплоскость, определяемая неформализованным ограничением: $\bar{H} = \{x \mid \bar{\varphi}(x) = 0\}$	1.2

ЛПФ	Задача линейного программирования с формализованной частью: $\max \{(c, x) \mid x \in D\}$	1.2
$\hat{x}$	Решение задачи ЛПФ	1.2
ЛПД	Дополнительная задача линейного программирования: $\min \{(c, x) \mid Ax \leq b, \tilde{\varphi}(x) \geq 0\}$	2.5
$\tilde{x}$	Решение задачи ЛПД	2.5
ЛП-ДА	Алгоритм решения задачи ЛПНО, базирующийся на синтезе методов линейного программирования и дискриминантного анализа	2