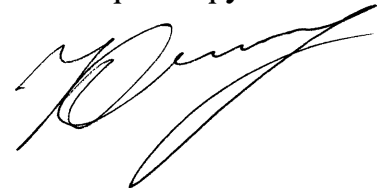


Федеральное государственное автономное образовательное учреждение
высшего образования
«ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(национальный исследовательский университет)»

На правах рукописи



ОЛЬХОВСКИЙ Николай Александрович

**МОДЕЛИ, МЕТОДЫ И АЛГОРИТМЫ ДЛЯ РЕШЕНИЯ
ЗАДАЧ ЛИНЕЙНОЙ ОПТИМИЗАЦИИ НА ОСНОВЕ
СИНТЕЗА СУПЕРКОМПЬЮТЕРНЫХ
И НЕЙРОСЕТЕВЫХ ТЕХНОЛОГИЙ**

Специальность 1.2.3. Теоретическая информатика, кибернетика

Диссертация на соискание ученой степени
кандидата физико-математических наук

Научный руководитель:
СОКОЛИНСКИЙ Леонид Борисович,
доктор физ.-мат. наук, профессор

Челябинск – 2023

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	3
1. АПЕКС-МЕТОД	8
1.1 Обзор методов решения задачи линейного программирования	8
1.2 Теоретические основы	16
1.3 Описание метода Апекс	25
1.3.1 Алгоритм вычисления псевдопроекции	26
1.3.2 Этап поиска	33
1.3.3 Целевой этап	34
1.4 Внедрение и вычислительные эксперименты	40
2. ВИЗУАЛЬНОЕ ПРЕДСТАВЛЕНИЕ n-МЕРНЫХ МНОГОГРАННИКОВ	42
2.1 Математическая модель визуального представления задачи ЛП	42
2.2 Параллельный алгоритм построения образа задачи ЛП	56
2.3 Вычислительные эксперименты	63
3. МЕТОД ПОВЕРХНОСТНОГО ДВИЖЕНИЯ	66
3.1 Теоретический базис	66
3.2 Метод поверхностного движения	76
3.3 Крестообразное рецептивное поле	83
3.4 Построение нейронной сети для движения по гиперплоскости	88
3.5 Вычислительные эксперименты	93
ЗАКЛЮЧЕНИЕ	96
ЛИТЕРАТУРА	98

ВВЕДЕНИЕ

Актуальность темы. Быстрое развитие технологий сбора и обработки больших данных [1; 2] привело к возникновению математических моделей оптимизации в форме многомерных задач линейного программирования [3]. Такие проблемы встречаются в промышленности, экономике, логистике, статистике, квантовой физике и других областях [4—6]. Важным классом прикладных задач ЛП являются нестационарные задачи, связанные с оптимизацией в динамических средах [7]. Для нестационарной задачи ЛП целевая функция и/или ограничения могут изменяться в процессе вычислений. Примеры нестационарных задач следующие: поддержка принятия решений в высокочастотной биржевой торговле [8; 9], гидро-газо-динамические задачи [10], оптимальное управление технологическими процессами [11—13], транспортом [14—16], ресурсами [17; 18].

Один из стандартных подходов к решению нестационарных задач оптимизации заключается в том, чтобы каждое изменение рассматривать в качестве новой задачи оптимизации, которую нужно решить с чистого листа [7]. Однако, такой подход зачастую не применим на практике, поскольку решение задачи с чистого листа без использования уже имеющейся информации может занимать слишком продолжительное время. Таким образом, желательно иметь оптимизационный алгоритм, способный постоянно подстраивать решение под изменения окружающей среды, используя данные, полученные на предыдущих шагах. Этот подход применим к процессам реального времени, если алгоритм достаточно быстро определяет траекторию перемещающейся оптимальной точки. В случае задач ЛП большой размерности последнее требование делает необходимой разработку масштабируемых методов и параллельных алгоритмов решения задач ЛП.

До настоящего времени одним из наиболее распространенных способов решения задачи ЛП был класс алгоритмов предложенных

и разработанных Данцигом, основанный на симплекс-методе [19]. Симплекс-метод был признан эффективным для решения большого класса задач ЛП. В частности, симплекс-метод легко преодолевают проблемы, связанные с большой размерностью в задаче ЛП [20]. Однако симплекс-метод имеет фундаментальные особенности, ограничивающие его применение для решения больших задач ЛП. Во-первых, в определенных случаях симплекс-метод вынужден перебирать все вершины симплекса, что ведет к экспоненциальному росту временных затрат [21—23]. Во-вторых, в большинстве случаев симплекс метод успешно решает задачи ЛП, содержащие до 50 000 переменных. Однако при решении больших задач ЛП наблюдается потеря точности [24]. Указанная потеря точности не может быть компенсирована даже за счет применения таких вычислительно затратных процедур, как «афинное преобразование» или «пошаговое уточнение» [25]. В-третьих, в общем случае последовательная природа симплекс-метода делает крайне сложным распараллеливание вычислений на мультипроцессорных системах с распределенной памятью [26]. Были предприняты многочисленные попытки построить масштабируемую параллельную реализацию симплекс-метода, но все они оказались безуспешными [27]. Во всех случаях граница масштабируемости была от 16 до 32 вычислительных узлов (см. напр. [28]). Хачиян доказал [29], что использование варианта метода эллипсоидов (предложен в 1970-х Шором [30], а также Юдиным и Немировским [31]) позволяет решать задачи ЛП за полиномиальное время. Однако попытки применить этот подход на практике были безуспешны, так как в подавляющем большинстве случаев метод эллипсоидов демонстрировал гораздо более низкую эффективность по сравнению с симплекс-методом. Позже Кармаркар [32] предложил алгоритм внутренних точек с полиномиальным временем, который мог быть использован на практике. Этот алгоритм стал базой для целой группы современных методов внутренних точек [33], которые способны решать задачи ЛП большой размерности с миллионами

переменных и миллионами уравнений [34—38]. Более того, эти методы являются самокорректирующимися, и таким образом обеспечивают высокую точность вычислений. Основным недостатком методов внутренних точек – это необходимость находить некоторую допустимую точку, удовлетворяющую всем ограничениям задачи ЛП перед началом вычислений. Нахождение такой внутренней точки может быть сведено к решению дополнительной задачи ЛП [39]. Другой метод для нахождения внутренней точки – метод псевдо-проекций [40], использующий фейеровские отображения [41]. Единственный наиболее существенный недостаток метода внутренних точек заключается в его плохой масштабируемости на многопроцессорных системах с распределенной памятью. Существует несколько успешных параллельных реализаций метода внутренних точек для отдельных случаев (см. напр. [42]), но в общем случае эффективная параллельная реализация на многопроцессорных системах построена быть не может. В соответствии с этим, разработка и исследование новых подходов к решению многомерных нестационарных задач ЛП в режиме реального времени является насущной проблемой.

Одним из наиболее обещающих подходов к решению сложных задач в режиме реального времени является использование нейросетевых моделей [43]. Искусственные нейронные сети (ИНН) – мощный универсальный инструмент, применяемый при решении прикладных задач практически во всех сферах. Наиболее известной нейросетевой моделью является сеть прямого распространения. Обучение и работа такой сети может быть очень эффективно реализована на графических ускорителях (GPU) [44]. Важным свойством сети прямого распространения является то, что время решения задачи – это известная константа, которая не зависит от параметров задачи. Эта особенность критически важна для режима реального времени. Впервые применение нейронных сетей к решению задач ЛП было описано в работе Танка и Хопфилда [45]. Была создана двухслойная рекуррентная нейронная сеть. Число нейронов в первом слое соответствовало числу

переменных в задаче ЛП. Число нейронов во втором слое совпадало с количеством ограничений задачи ЛП. Первый и второй слой были сильно связанными. Веса и смещения однозначно определялись коэффициентами и правыми частями линейных неравенств, задаваемых ограничениями, а также коэффициентами линейной целевой функции. Состояние нейронной сети определялось дифференциальным уравнением $\dot{x}(t) = \nabla E(x(t))$, где $E(x(t))$ является энергетической функцией специального типа. Вначале произвольная точка допустимой области поступала на вход нейронной сети. Затем сигналы второго слоя рекурсивно передавались на первый. Этот процесс сходил к стабильному состоянию, когда выходные остаются неизменными. Это состояние соответствовало минимуму энергетической функции, а выходной сигнал - решению задачи ЛП. Метод Танка и Хопфилда был развит и дополнен в последовавших работах (см. напр. [46—50]). Главным недостатком такого подхода является непредсказуемое число рабочих циклов нейронной сети. Таким образом, рекуррентная сеть, основанная на энергетической функции не может быть использована для решения задач ЛП большой размерности в реальном времени.

В недавнем исследовании [51] была предложена n -мерная математическая модель визуализации задач ЛП. Эта модель сделала возможным применение сетей прямого распространения, включая сверточные нейронные сети [52], для решения многомерных задач ЛП, допустимая область решений которых представляет собой ограниченное непустое множество. В настоящее время в научных изданиях практически не встречаются работы, посвященные использованию сверточных нейронных сетей для решения задач ЛП [53]. Причина заключается в том, что сверточные нейронные сети предназначены главным образом для обработки изображений, но на сегодняшний день не существует методов, позволяющих сгенерировать массив обучающих данных, основанный на визуальном представлении многомерных задач ЛП.

Целью данной работы является разработка моделей, методов и алгоритмов решения многомерных задач линейного программирования на основе синтеза суперкомпьютерных и нейросетевых технологий. Для достижения данной цели были сформулированы следующие **задачи**.

1. Исследовать имеющиеся итерационные методы решения задачи ЛП. Разработать на их основе метод решения многомерной задачи ЛП, подходящий для обучения искусственных нейронных сетей.
2. Разработать метод визуализации многомерных задач ЛП. Реализовать алгоритм параллельных вычислений на его основе, обладающий высокой масштабируемостью.
3. Разработать итерационный метод решения многомерных задач ЛП на основе комбинации визуального представления задачи и нейронной сети прямого распространения.

Структура и содержание работы. Диссертация состоит из введения, 3 глав и заключения. Полный объём диссертации составляет 108 страниц, включая 21 рисунок и 11 таблиц. Список литературы содержит 88 наименований.

В первой главе приводится обзор итерационных методов решения задач ЛП. Описывается новый метод для решения задач высокой размерности. В конце главы дается оценка сложности реализованного алгоритма и результаты вычислительных экспериментов.

Во второй главе описывается метод визуализации задач ЛП высокой размерности. Демонстрируется параллельный алгоритм вычислений, а также рамочный алгоритм взаимодействия вычислителя и искусственной нейронной сети. В конце главы дается оценка сложности реализованного алгоритма и результаты вычислительных экспериментов.

В третьей главе описывается обобщенный итерационный метод решения многомерной задачи ЛП на основе метода визуализации. Дается пошаговый алгоритм и доказывается его сходимость к точному решению задачи.

1. АПЕКС-МЕТОД

1.1 Обзор методов решения задачи линейного программирования

В данном разделе приведен обзор работ, посвященных итерационным проективным методам, используемым для нахождения допустимой точки и решения задач ЛП. Задача о нахождении допустимого значения задачи ЛП определяется следующим образом. Рассмотрим систему линейных неравенств в матричной форме

$$Ax \leq b, \quad (1.1)$$

где $A \in \mathbb{R}^{m \times n}$, и $b \in \mathbb{R}^n$. Во избежание тривиальности, мы полагаем $m > 1$. Задача о нахождении допустимой точки заключается в определении такой точки $\tilde{x} \in \mathbb{R}^n$, которая будет удовлетворять матричной системе неравенств (1.1). Здесь и далее мы полагаем, что такая точка существует.

Проективные методы полагаются на следующую геометрическую интерпретацию области допустимых значений задачи ЛП. Пусть $a_i \in \mathbb{R}^n$ является вектором, составленным из элементов i -той строки матрицы A . Тогда матрица неравенств $Ax \leq b$ может быть представлена в виде системы неравенств

$$\langle a_i, x \rangle \leq b_i, i = 1, \dots, m. \quad (1.2)$$

Здесь и далее $\langle \cdot, \cdot \rangle$ обозначено скалярное произведение векторов. Мы полагаем с этого момента, что

$$a_i \neq 0 \quad (1.3)$$

для всех $i = 1, \dots, m$. Для каждого неравенства $\langle a_i, x \rangle \leq b_i$ определим замкнутое полупространство

$$\hat{H}_i = \{x \in \mathbb{R}^n | \langle a_i, x \rangle \leq b_i\}, \quad (1.4)$$

и его граничную гиперплоскость

$$H_i = \{x \in \mathbb{R}^n | \langle a_i, x \rangle = b_i\}. \quad (1.5)$$

Для любой точки $x \in \mathbb{R}^n$, ортогональная проекция $\pi(x)$ точки x на гиперплоскость H_i может быть вычислена по формуле

$$\pi_i(x) = x - \frac{\langle a_i, x \rangle - b_i}{\|a_i\|^2} a_i. \quad (1.6)$$

Здесь и далее $\|\cdot\|$ обозначена Евклидова норма. Определим

$$M = \bigcap_{i=1}^m \hat{H}_i. \quad (1.7)$$

Мы предполагаем, что $M \neq \emptyset$, т.е. решение системы (1.1) существует. В этом случае, M является выпуклым замкнутым многогранником в пространстве \mathbb{R}^n . В геометрической интерпретации задача о нахождении допустимого решения заключается в определении точки $\tilde{x} \in M$.

Родоначалниками итерационных проективных методов определения допустимого решения являются Качмаж и Чиммино. В [54] (перевод на английский доступен в [55]), Качмаж представил метод последовательных проекций для решения совместной системы линейных уравнений

$$\langle a_i, x \rangle = b_i, i = 1, \dots, m. \quad (1.8)$$

Его метод, начиная с произвольной точки $x^{(0,m)} \in \mathbb{R}^n$, вычисляет дальнейшую последовательность групп точек:

$$x^{(k,1)} = \pi_1 \left(x^{(k-1,m)} \right), x^{(k,2)} = \pi_2 \left(x^{(k,1)} \right), \dots, x^{(k,m)} = \pi_m \left(x^{(k,m-1)} \right) \quad (1.9)$$

для $k = 1, 2, 3, \dots$. Здесь π_i ($i = 1, \dots, m$) обозначает ортогональную проекцию на гиперплоскость H_i , заданную уравнением (1.6). Эта последовательность сходится к решению системы (1.8). Геометрически метод может быть интерпретирован следующим образом. Строится ортогональная проекция начальной точки $x^{(0,m)}$ на гиперплоскость H_1 . Проекция является точкой $x^{(1,1)}$, которая теперь проектируется на H_2 . Получившаяся точка $x^{(1,2)}$ проектируется на H_3 , что дает точку $x^{(1,3)}$ и т.д. В результате мы получаем последнюю точку $x^{(1,m)}$ из первой группы точек. Вторая группа

точек строится аналогичным образом, начиная с точки $x^{(1,m)}$. Процесс повторяется для $k = 2, 3, \dots$

Чиммино предложил в [56] (описание на английском в [57]) метод синхронного проектирования для одной задачи. В этом методе используется следующая операция *ортогонального отражения*

$$\rho_i(x) = x - 2 \frac{\langle a_i, x \rangle - b_i}{\|a_i\|^2} a_i, \quad (1.10)$$

которая вычисляет точку $\rho_i(x)$ симметрично точке x относительно гиперплоскости H_i . Для текущего приближения $x^{(k)}$, метод Чиммино синхронно вычисляет отражения относительно всех гиперплоскостей H_i ($i = 1, \dots, m$), а затем выпуклая комбинация этих отражений используется, чтобы сформировать следующее приближение:

$$x^{(k+1)} = \sum_{i=1}^m w_i \rho_i(x^{(k)}), \quad (1.11)$$

где $w_i > 0$ ($i = 1, \dots, m$), $\sum_{i=1}^m w_i = 1$. Когда $w_i = \frac{1}{m}$ ($i = 1, \dots, m$), уравнение (1.11) преобразуется в следующую формулу:

$$x^{(k+1)} = \frac{1}{m} \sum_{i=1}^m \rho_i(x^{(k)}). \quad (1.12)$$

Агмон [58] и Мотцкин и Шенберг [59] обобщили метод проекции с уравнений на неравенства. Для решения задачи (1.1), они ввели *проекцию релаксации*

$$\pi_i^\lambda(x) = (1 - \lambda)x + \lambda \pi_i(x), \quad (1.13)$$

где $0 < \lambda < 2$. Очевидно, что $\pi_i^1(x) = \pi_i(x)$. Для вычисления следующего приближения, в методе проекции релаксации используется следующее уравнение:

$$x^{(k+1)} = \pi_l^\lambda(x^{(k)}), \quad (1.14)$$

где

$$\|x^{(k)} - \pi_l(x^{(k)})\| = \max_{x^{(k)} \notin \hat{H}_i} \|x^{(k)} - \pi_i(x^{(k)})\|. \quad (1.15)$$

Неформально, следующее приближение $x^{(k+1)}$ является проекцией релаксации предыдущего приближения $x^{(k)}$ относительно наиболее удаленной гиперплоскости H_l , ограничивающей полупространство \hat{H}_l , не содержащее $x^{(k)}$. Агмон в [58] показал, что последовательность $x^{(k)}$ сходится, как $k \rightarrow \infty$, к точке на границе of M .

Сенсор и Элфайинг в [60], обобщили метод Чиммино на случай линейных неравенств. Они определили проекцию релаксации на полупространство \hat{H}_i следующим образом:

$$\hat{\pi}_i^\lambda(x) = (1 - \lambda)x - \lambda \frac{\max\{0, \langle a_i, x \rangle - b_i\}}{\|a_i\|^2} a_i \quad (1.16)$$

что дает уравнение

$$x^{(k+1)} = \sum_{i=1}^m w_i \hat{\pi}_i^\lambda(x^{(k)}). \quad (1.17)$$

Здесь $0 < \lambda < 2$ и $w_i > 0$ ($i = 1, \dots, m$), $\sum_{i=1}^m w_i = 1$. В [61], Де Пьерро предложил способ доказательства сходимости этого метода, который отличается от подхода Сенсора и Элфайинга. Доказательство Де-Пьерро также приемлемо для случая, когда исходная система линейных неравенств не имеет решений. В этом случае для $\lambda = 1$, последовательность (1.17) сходится к точке минимума функции $f(x) = \sum_{i=1}^m w_i \|\hat{\pi}_i(x) - x\|^2$, т.е. является решением по методу наименьших взвешенных (с весами w_i) квадратов системы (1.1).

Методы, подобные предложенному Чиммино, допускают эффективное распараллеливание, поскольку ортогональные проекции (отражения) могут вычисляться одновременно и независимо друг от друга. В статье [62] исследуется эффективность распараллеливания семейства методов Чиммино на многоядерных процессорах Xeon Phi. В [63] приведена оценка масштабируемости метода Чиммино для многопроцессорных систем с распределенной памятью. Применение семейства методов Чиммино для решения нестационарных систем линейных неравенств на вычислительных кластерах рассмотрено в [64].

В качестве последней работы стоит отметить статью [65], которая распространяет метод релаксации Агмона-Мотцкина-Шенберга на случай полу-бесконечных системы неравенств. Авторы рассматривают систему с бесконечным числом неравенств в конечномерном Евклидовом пространстве \mathbb{R}^n :

$$\langle a_i, x \rangle \leq b_i, i \in I, \quad (1.18)$$

где I – произвольный бесконечный набор индексов. Основная идея метода следующая. Пусть гиперплоскость $H_x^{(\infty)} = \sup \{ \max (\langle a_i, x \rangle - b_i, 0) \mid i \in I \}$ будет *наибольшим изменением* по отношению к x . Пусть $x^{(0)}$ – произвольная начальная точка. Если текущая итерация $x^{(k)}$ не является системой (1.18), то пусть $x^{(k+1)}$ – ортогональная проекция $x^{(k)}$ на гиперплоскость $H_i (i \in I)$ в окрестности наибольшего изменения $H_{x^{(k)}}^{(\infty)}$. Если система (1.18) имеет решение, тогда последовательность $\{x^{(k)} \mid k = 1, 2, \dots\}$ сформированная описанным способом, сходится к решению этой системы.

Решение систем линейных неравенств имеет непосредственной отношение к задачам ЛП, поэтому проективные методы могут быть эффективно использованы для решения этого класса задач. Эквивалентность задачи о нахождении допустимого решения и задачи ЛП основывается двойственной задаче ЛП. Рассмотрим прямую задачу ЛП в матричной форме:

$$\bar{x} = \arg \max_x \{ \langle c, x \rangle \mid Ax \leq b, x \geq \mathbf{0} \}, \quad (1.19)$$

где $c, x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$, and $c \neq \mathbf{0}$. Здесь и далее, $\langle \cdot, \cdot \rangle$ обозначено скалярное произведение векторов. Построим двойственную задачу по отношению к задаче (1.19):

$$\bar{u} = \arg \min_u \{ \langle b, u \rangle \mid A^T u \geq c, u \geq \mathbf{0} \}, \quad (1.20)$$

где $u \in \mathbb{R}^m$. Возникает следующее прямо-двойственное уравнение:

$$\langle c, \bar{x} \rangle = \max_{Ax \leq b, x \geq \mathbf{0}} \langle c, x \rangle = \min_{A^T u \geq c, u \geq \mathbf{0}} \langle b, u \rangle = \langle b, \bar{u} \rangle. \quad (1.21)$$

В [66; 67] Еремин предложил следующий метод, основанный на прямо-двойственном подходе. Пусть система неравенств

$$A'x \leq b' \quad (1.22)$$

определяет область допустимых значений прямой задачи (1.19). Эта система получается путем добавления к системе $Ax \leq b$ вектора неравенства $-x \leq 0$. В этом случае $A' \in \mathbb{R}^{(m+n) \times n}$ и $b' \in \mathbb{R}^{m+n}$. Пусть a'_i обозначена i -ая строка матрицы A' . Для каждого неравенства $\langle a'_i, x \rangle \leq b'_i$ определим замкнутое полупространство

$$\hat{H}'_i = \{x \in \mathbb{R}^n | \langle a'_i, x \rangle \leq b'_i\}, \quad (1.23)$$

и его ограничивающую гиперплоскость

$$H'_i = \{x \in \mathbb{R}^n | \langle a'_i, x \rangle = b'_i\}. \quad (1.24)$$

Пусть $\pi'_i(x)$ обозначена ортогональная проекция точки x на гиперплоскость H'_i :

$$\pi'_i(x) = x - \frac{\langle a'_i, x \rangle - b'_i}{\|a'_i\|^2} a'_i. \quad (1.25)$$

Определим проекцию на гиперплоскость \hat{H}'_i :

$$\hat{\pi}'_i(x) = x - \frac{\max\{0, \langle a'_i, x \rangle - b'_i\}}{\|a'_i\|^2} a'_i. \quad (1.26)$$

Эта проекция обладает следующими двумя свойствами:

$$x \notin \hat{H}'_i \Rightarrow \hat{\pi}'_i(x) = \pi'_i(x); \quad (1.27)$$

$$x \in \hat{H}'_i \Rightarrow \hat{\pi}'_i(x) = x. \quad (1.28)$$

Определим $\varphi_1 : \mathbb{R}^n \rightarrow \mathbb{R}^n$ следующим образом:

$$\varphi_1(x) = \frac{1}{m+n} \sum_{i=1}^{m+n} \hat{\pi}'_i(x). \quad (1.29)$$

Аналогично определим область допустимых значений двойственной задачи (1.20) как

$$D'x \geq c', \quad (1.30)$$

где $D = A^T \in \mathbb{R}^{n \times m}$, $D' \in \mathbb{R}^{(m+n) \times m}$, and $c' \in \mathbb{R}^{n+m}$. Обозначим

$$\hat{\eta}'_j(u) = u - \frac{\max\{0, \langle d'_j, u \rangle - c'_j\}}{\|d'_j\|^2} d'_j, \quad (1.31)$$

и определим $\varphi_2 : \mathbb{R}^m \rightarrow \mathbb{R}^m$ как

$$\varphi_2(u) = \frac{1}{n+m} \sum_{j=1}^{n+m} \hat{\eta}'_j(x). \quad (1.32)$$

Теперь определим $\varphi_3 : \mathbb{R}^{n+m} \rightarrow \mathbb{R}^{n+m}$ как

$$\varphi_3([x, u]) = [x, u] - \frac{\langle c, x \rangle - \langle b, u \rangle}{\|c\|^2 + \|b\|^2} [c, -b], \quad (1.33)$$

что соответствует уравнению (1.21). Здесь $[\cdot, \cdot]$ обозначена конкатенация векторов.

Наконец, определим $\varphi : \mathbb{R}^{n+m} \rightarrow \mathbb{R}^{n+m}$ следующим образом:

$$\varphi([x, u]) = \varphi_3(\varphi_1(x), \varphi_2(u)). \quad (1.34)$$

Если область допустимых решений прямой задачи является ограниченным непустым множеством, то последовательность

$$\left[x^{(k+1)}, u^{(k+1)} \right] = \varphi \left(\left[x^{(k)}, u^{(k)} \right] \right) \quad (1.35)$$

сходится к точке $[\bar{x}, \bar{u}]$, где \bar{x} – решение прямой задачи (1.19), а \bar{u} – решение двойственной задачи (1.20).

В статье [68] предлагается метод решения невырожденных задач ЛП, основанный на вычислении ортогональной проекции некоторой специальной точки, не зависящей от основной части данных, описывающих задачу ЛП, на зависящий от постановки задачи конус, образованный ограничивающими неравенствами. В действительности, этот метод решает

симметричную положительно определенную систему линейных уравнений специального вида. Автор описывает конечный алгоритм из семейства активных множеств, который способен вычислять ортогональные проекции для задач, содержащих тысячи строк и столбцов. Основным недостатком предложенного метода - это существенное увеличение размерности задачи, по сравнению с исходными данными.

В статье [69] Сенсор предлагает метод линейной супериоризации (LinSup) в качестве инструмента для решения задач ЛП. Использование метода LinSup не гарантирует нахождение точки минимума задачи ЛП, но он направляет алгоритм поиска линейной зависимости к точке с уменьшающимся значением целевой функции. Этот процесс не идентичен тому, что используется в решателях задач ЛП, но он может быть альтернативой Симплекс-методу для задач большого размера. Основная идея LinSup в том, чтобы добавить дополнительное условие, называемое *параметр возмущения*, в итерационное уравнение проективного метода. Параметр возмущения направляет алгоритм поиска допустимого решения по направлению к резкому уменьшению целевой функции. В случае задачи ЛП (1.19), целевой функцией является $f(x) = \langle c, x \rangle$, и LinSup добавляет $\left(-\eta \frac{c}{\|c\|}\right)$ в качестве параметра возмущения к итерационному уравнению (1.17):

$$x^{(k+1)} = \left(-\eta \frac{c}{\|c\|}\right) + \sum_{i=1}^m w_i \hat{\pi}_i^\lambda \left(x^{(k)}\right). \quad (1.36)$$

Здесь $0 < \eta < 1$ обозначен *параметр возмущения*.

Статья [51] предлагает математическую модель для визуального представления многомерных задач ЛП, вводит понятие целевой гиперплоскости H_c , нормаль к которой образует градиент целевой функции $f(x) = \langle c, x \rangle$. В ситуации поиска максимума целевая гиперплоскость располагается так, чтобы значение целевой функции во всех ее точках было больше, чем значение целевой функции в любой точке выпуклого многогранника M , представляющего из себя область допустимых значений

задачи ЛП. Для любой точки $g \in H_c$, целевая проекция $\gamma_M(g)$ на M определяется следующим образом:

$$\gamma_M(g) = \begin{cases} \arg \min_x \{ \|x - g\| \mid x \in M, \pi_{H_c}(x) = g \}, & \text{if } \exists x \in M : \pi_{H_c}(x) = g; \\ +\infty, & \text{if } \neg \exists x \in M : \pi_{H_c}(x) = g. \end{cases} \quad (1.37)$$

Здесь $\pi_{H_c}(x)$ обозначена ортогональная проекция на H_c . На целевой гиперплоскости H_c строится прямоугольное поле точек $\mathfrak{G} \in \mathbb{R}^n \times \mathbb{R}^{K(n-1)}$, где K обозначено число точек по каждому измерению. Каждая точка $g \in \mathfrak{G}$ отображается в действительное число $\|\gamma_M(g) - g\|$. Это отображение формирует матрицу размерности $(n-1)$, которая является *образом задачи ЛП*. Такой подход открывает возможности использования искусственных нейронных сетей прямого распространения, включая сверточные нейронные сети, для решения многомерных задач ЛП. Одно из главных препятствий для реализации такого подхода – это проблема генерации обучающего множества. Обзор научной литературы показал, что не существует метода, способного построить обучающее множество, совместимое с описанным подходом. В последующих разделах представлен такой метод.

1.2 Теоретические основы

В этом разделе мы представляем теоретическую основу, используемую для построения Апекс-метода. Рассмотрим задачу ЛП в следующей форме:

$$\bar{x} = \arg \max_{x \in \mathbb{R}^n} \{ \langle c, x \rangle \mid Ax \leq b \}, \quad (1.38)$$

где $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$, $m > 1$, и $c \neq \mathbf{0}$. Мы предполагаем, что ограничение $x \geq \mathbf{0}$ также включено в систему $Ax \leq b$ в виде следую-

щих неравенств:

$$-x_1 \leq 0;$$

...

$$-x_n \leq 0.$$

Пусть \mathcal{P} обозначает множество индексов строк в матрице A :

$$\mathcal{P} = \{1, \dots, m\}. \quad (1.39)$$

Пусть $a_i \in \mathbb{R}^n$ – вектор, образованный элементами i -ой строки матрицы A , и $a_i \neq \mathbf{0}$ для всех $i \in \mathcal{P}$. Обозначим через \hat{H}_i замкнутое полупространство, определяемое неравенством $\langle a_i, x \rangle \leq b_i$, а через H_i гиперплоскость, ограничивающую \hat{H}_i :

$$\hat{H}_i = \{x \in \mathbb{R}^n \mid \langle a_i, x \rangle \leq b_i\}; \quad (1.40)$$

$$H_i = \{x \in \mathbb{R}^n \mid \langle a_i, x \rangle = b_i\}. \quad (1.41)$$

Определение 1.1. Полупространство \hat{H} называется доминантным относительно вектора c , или кратко c -доминирующим, если

$$\forall x \in \hat{H}, \forall \lambda \in \mathbb{R}_{>0} : x + \lambda c \in \hat{H}. \quad (1.42)$$

Геометрический смысл этого определения состоит в том, что луч, исходящий из точки, принадлежащей полупространству, в направлении вектора c принадлежит этому полупространству.

Определение 1.2. Полупространство \hat{H} называется рецессивным по отношению к вектору c , или кратко c -рецессивным, если оно не является c -доминантным, т.е.,

$$\forall x \in \hat{H}, \exists \lambda \in \mathbb{R}_{>0} : x + \lambda c \notin \hat{H}. \quad (1.43)$$

Следующее предложение дает необходимое и достаточное условие для c -рецессивности полупространства.

Утверждение 1.3. Пусть полупространство \hat{H} определяется следующим уравнением:

$$\hat{H} = \{x \in \mathbb{R}^n \mid \langle a, x \rangle \leq \beta\}. \quad (1.44)$$

Тогда необходимым и достаточным условием для c -рецессивности полупространства \hat{H} является

$$\langle a, c \rangle > 0. \quad (1.45)$$

Доказательство. Сначала докажем необходимость. Пусть условие (3.34) имеет место. Обозначим

$$x' = \frac{\beta a}{\|a\|^2}. \quad (1.46)$$

Из этого следует

$$\langle a, x' \rangle = \left\langle a, \frac{\beta a}{\|a\|^2} \right\rangle = \beta \frac{\langle a, a \rangle}{\|a\|^2} = \beta, \quad (1.47)$$

т.е. $x' \in \hat{H}$. В силу (3.34), существует $\lambda' \in \mathbb{R}_{>0}$ такой, что

$$x' + \lambda' c \notin \hat{H}, \quad (1.48)$$

т.е.,

$$\langle a, x' + \lambda' c \rangle > \beta. \quad (1.49)$$

Подставляя правую часть уравнения (1.46) вместо x' , получаем

$$\left\langle a, \frac{\beta a}{\|a\|^2} + \lambda' c \right\rangle > \beta. \quad (1.50)$$

Поскольку $\lambda' > 0$, из этого следует, что

$$\langle a, c \rangle > 0. \quad (1.51)$$

Таким образом, необходимость доказана.

Докажем достаточность от противного. Предположим, что (3.35) выполняется, и \hat{H} не является c -рецессивным, т.е.,

$$\forall x \in \hat{H}, \forall \lambda \in \mathbb{R}_{>0} : x + \lambda c \in \hat{H}. \quad (1.52)$$

Так как x' , определяемый (1.46), принадлежит \hat{H} , то из этого следует.

$$x' + \lambda c \in \hat{H} \quad (1.53)$$

для всех $\lambda \in \mathbb{R}_{>0}$, т.е.

$$\langle a, x' + \lambda c \rangle \leq \beta. \quad (1.54)$$

Подставляя правую часть уравнения (1.46) вместо x' , получаем

$$\left\langle a, \frac{\beta a}{\|a\|^2} + \lambda c \right\rangle \leq \beta. \quad (1.55)$$

Поскольку $\lambda > 0$, из этого следует, что

$$\langle a, c \rangle \leq 0. \quad (1.56)$$

Но это противоречит (3.35).

Обозначим

$$e_c = \frac{c}{\|c\|}, \quad (1.57)$$

т.е. e_c обозначает единичный вектор, параллельный вектору c .

Утверждение 1.4. Пусть полупространство \hat{H}_i является c -рецессивным.

Тогда для любой точки $x' \in \mathbb{R}$ и любого числа $\eta > 0$, точка

$$z = x' + \left(\eta + \frac{b_i - \langle a_i, x' \rangle}{\langle a_i, e_c \rangle} \right) e_c \quad (1.58)$$

не принадлежит полупространству \hat{H}_i , т.е.

$$\langle a_i, z \rangle > b_i. \quad (1.59)$$

Доказательство. Полупространство \hat{H}_i является c -рецессивным, поэтому, согласно Предложению 1.3, имеет место следующее неравенство:

$$\langle a_i, c \rangle > 0. \quad (1.60)$$

Принимая во внимание (1.58), мы имеем

$$\langle a_i, z \rangle = \left\langle a_i, x' + \left(\eta + \frac{b_i - \langle a_i, x' \rangle}{\langle a_i, e_c \rangle} \right) e_c \right\rangle = \eta \langle a_i, e_c \rangle + b_i. \quad (1.61)$$

Подставляя правую часть уравнения (3.25) вместо e_c в (1.61), получаем

$$\langle a_i, z \rangle = \frac{\eta}{\|c\|} \langle a_i, c \rangle + b_i. \quad (1.62)$$

Поскольку $\eta > 1$, в силу (1.60), неравенство $\frac{\eta}{\|c\|} \langle a_i, c \rangle > 0$ выполняется. Отсюда следует, что $\langle a_i, z \rangle > b_i$, т.е. $z \notin \hat{H}_i$.

Определим

$$\mathcal{I}_c = \{i \in \mathcal{P} \mid \langle a_i, c \rangle > 0\}, \quad (1.63)$$

т.е. \mathcal{I}_c – это множество индексов, для которых полупространство \hat{H}_i является c -рецессивным. Здесь и далее полагаем, что

$$\mathcal{I}_c \neq \emptyset. \quad (1.64)$$

Утверждение 1.5. Пусть дана произвольная допустимая точка x' задачи ЛП (3.1):

$$\forall i \in \mathcal{P} : \langle a_i, x' \rangle \leq b_i. \quad (1.65)$$

Тогда, для любого положительного числа $\eta \in \mathbb{R}_{>0}$, точка

$$z = x' + \left(\eta + \max \left\{ \frac{b_i - \langle a_i, x' \rangle}{\langle a_i, e_c \rangle} \mid i \in \mathcal{I}_c \right\} \right) e_c \quad (1.66)$$

не принадлежит ни одному c -рецессивному полупространству \hat{H}_i , т.е.,

$$\forall i \in \mathcal{I}_c : \langle a_i, z \rangle > b_i. \quad (1.67)$$

Доказательство. Из (1.65), получаем

$$\forall i \in \mathcal{I}_c : b_i - \langle a_i, x' \rangle \geq 0. \quad (1.68)$$

Согласно (3.40) и (3.25), выполняется следующее условие:

$$\forall i \in \mathcal{I}_c : \langle a_i, e_c \rangle > 0. \quad (1.69)$$

Следовательно,

$$\max \left\{ \frac{b_i - \langle a_i, x' \rangle}{\langle a_i, e_c \rangle} \mid i \in \mathcal{I}_c \right\} \geq 0 \quad (1.70)$$

для любого $i \in \mathcal{I}_c$. Зафиксируем любое $j \in \mathcal{I}_c$, и определим

$$\eta' = \eta + \max \left\{ \frac{b_i - \langle a_i, x' \rangle}{\langle a_i, e_c \rangle} \mid i \in \mathcal{I}_c \right\} - \frac{b_j - \langle a_j, x' \rangle}{\langle a_j, e_c \rangle}, \quad (1.71)$$

где $\eta > 0$. Из (1.70), следует, что $\eta' > 0$. Используя (1.66) и (1.71), получаем

$$z = x' + \left(\eta + \max \left\{ \frac{b_i - \langle a_i, x' \rangle}{\langle a_i, e_c \rangle} \mid i \in \mathcal{I}_c \right\} \right) e_c = x' + \left(\eta' + \frac{b_j - \langle a_j, x' \rangle}{\langle a_j, e_c \rangle} \right) e_c. \quad (1.72)$$

Из предложения 1.4 следует, что $\langle a_j, z \rangle > b_j$, т.е. точка z , определяемая (1.66), не принадлежит полупространству \hat{H}_j для любого $j \in \mathcal{I}_c$.

Следующее предложение определяет область, содержащую решение задачи ЛП (3.1).

Утверждение 1.6. Пусть \bar{x} – решение задачи ЛП (3.1). Тогда существует индекс $i' \in \mathcal{I}_c$ такой, что

$$\bar{x} \in H_{i'}, \quad (1.73)$$

т.е. существует c -рецессивное полупространство $\hat{H}_{i'}$ такое, что его граничная гиперплоскость $H_{i'}$ включает \bar{x} .

Доказательство. Обозначим через \mathcal{J}_c множество индексов, для которых полупространство \hat{H}_j является c -доминантным:

$$\mathcal{J}_c = \mathcal{P} \setminus \mathcal{I}_c. \quad (1.74)$$

Так как \bar{x} принадлежит области допустимых значений задачи ЛП (3.1), то

$$\bar{x} \in \bigcap_{j \in \mathcal{J}_c} \hat{H}_j, \quad (1.75)$$

и

$$\bar{x} \in \bigcap_{i \in \mathcal{I}_c} \hat{H}_i. \quad (1.76)$$

Определим луч Y следующим образом:

$$Y = \{ \bar{x} + \lambda c \mid \lambda \in \mathbb{R}_{\geq 0} \}. \quad (1.77)$$

По определению 1.1, мы имеем

$$Y \subset \bigcap_{j \in \mathcal{J}_c} \hat{H}_j, \quad (1.78)$$

т.е. луч Y принадлежит всем c -доминантным полупространствам. В силу Определения 2.2,

$$\forall i \in \mathcal{I}_c, \exists \lambda \in \mathbb{R}_{>0} : \bar{x} + \lambda c \notin \hat{H}_i. \quad (1.79)$$

Принимая во внимание (1.76), получаем, что

$$i \in \mathcal{I}_c : \cap H_i = y_i \in \mathbb{R}^n, \quad (1.80)$$

т.е. пересечение луча Y и любой гиперплоскости H_i , ограничивающей c -рецессивное полупространство \hat{H}_i , является единственной точкой $y_i \in \mathbb{R}^n$.

Пусть

$$i' = \arg \min_{i \in \mathcal{I}_c} \{ \|\bar{x} - y_i\| \mid y_i = Y \cap H_i \}, \quad (1.81)$$

т.е. $H_{i'}$ – ближайшая гиперплоскость к точке \bar{x} для $i' \in \mathcal{I}_c$. Обозначим через \bar{y} пересечение луча Y и гиперплоскости $H_{i'}$:

$$\bar{y} = Y \cap H_{i'}. \quad (1.82)$$

Согласно (3.57),

$$\bar{y} \in \bigcap_{i \in \mathcal{I}_c} \hat{H}_i, \quad (1.83)$$

т.е. точка \bar{y} принадлежит всем c -рецессивным полупространствам.

Из (1.78), следует, что

$$\bar{y} \in \bigcap_{i \in \mathcal{P}} \hat{H}_i, \quad (1.84)$$

т.е. \bar{y} принадлежит области допустимых значений задачи ЛП (3.1). Пусть

$$\lambda' = \|\bar{x} - \bar{y}\|. \quad (1.85)$$

Тогда, в силу (1.77),

$$\langle c, \bar{y} \rangle = \langle c, \bar{x} + \lambda' e_c \rangle = \langle c, \bar{x} \rangle + \lambda' \frac{\langle c, c \rangle}{\|c\|} = \langle c, \bar{x} \rangle + \lambda' \|c\|. \quad (1.86)$$

Поскольку \bar{x} является решением задачи ЛП (3.1), то выполняется следующее условие:

$$\forall y \in \bigcap_{i \in \mathcal{P}} \hat{H}_i : \langle c, y \rangle \leq \langle c, \bar{x} \rangle. \quad (1.87)$$

Сравнивая это с (1.84), получаем, что

$$\langle c, \bar{y} \rangle \leq \langle c, \bar{x} \rangle. \quad (1.88)$$

Учитывая, что $\lambda' \geq 0$ и $c \neq 0$, в силу (1.86) и (1.88), получаем $\lambda' = 0$. Из (1.85) следует, что $\bar{x} = \bar{y}$. По (1.82), это означает, что $\bar{x} \in H_{\mathcal{V}}$, где $\hat{H}_{\mathcal{V}}$ – c -реcessивное полупространство.

Определение 1.7. Пусть $M \neq \emptyset$ – выпуклое замкнутое множество. Однозначное отображение $\varphi : \mathbb{R}^n \rightarrow \mathbb{R}^n$ называется M -Fejér отображением [66], если

$$x \in M : \varphi(x) = x, \quad (1.89)$$

и

$$\forall x \notin M, \forall y \in \mathbb{R}^n : \|\varphi(x) - y\| < \|x - y\|. \quad (1.90)$$

Утверждение 1.8. Пусть $x^{(0)} \in \mathbb{R}^n$. Если $\varphi(\cdot)$ – непрерывное отображение M -Fejér и

$$\left\{ x^{(k)} = \varphi^k \left(x^{(0)} \right) \right\}_{k=1}^{\infty}$$

является итерационным процессом, порожденным этим отображением, тогда

$$x^{(k)} \rightarrow \tilde{x} \in M. \quad (1.91)$$

Доказательство. Сходимость следует непосредственно из Теоремы 6.2 и Леммы 6.3 в [66].

Пусть $\pi_i(x)$ обозначает ортогональную проекцию точки x на гиперплоскость H_i :

$$\pi_i(x) = x - \frac{\langle a_i, x \rangle - b_i}{\|a_i\|^2} a_i. \quad (1.92)$$

Следующее предложение дает непрерывное отображение M -Fejér, которое будет использоваться в Апекс-методе.

Утверждение 1.9. Пусть $M \neq \emptyset$ – выпуклое замкнутое множество, представляющее область допустимых решений задачи ЛП (3.1):

$$M = \bigcap_{i=1}^m \hat{H}_i. \quad (1.93)$$

Для любой точки $x \in \mathbb{R}^n$ определим

$$\mathcal{J}_x = \{i \mid \langle a_i, x \rangle > b_i; i \in \mathcal{P}\}, \quad (1.94)$$

т.е. \mathcal{J}_x – это множество индексов, для которых полупространство \hat{H}_i не содержит точку x . Тогда однозначное отображение $\psi : \mathbb{R}^n \rightarrow \mathbb{R}^n$, определяемое уравнением

$$\psi(x) = \begin{cases} x, & \text{if } x \in M; \\ \frac{1}{|\mathcal{J}_x|} \sum_{i \in \mathcal{J}_x} \pi_i(x), & \text{if } x \notin M, \end{cases} \quad (1.95)$$

является непрерывным отображением M -Fejér.

Доказательство. Очевидно, что отображение $\psi(\cdot)$ является непрерывным. Докажем, что условие (1.90) выполняется. Наше доказательство основано на общей схеме, представленной в [66]. Пусть $y \in M$, а $x \notin M$. Отсюда следует, что

$$\mathcal{J}_x \neq \emptyset. \quad (1.96)$$

В силу уравнения (1.94), имеет место следующее неравенство

$$\|\pi_i(x) - x\| > 0 \quad (1.97)$$

для всех $i \in \mathcal{J}_x$. Согласно лемме 3.13 в [66], следующее неравенство имеет место для всех $i \in \mathcal{J}_x$:

$$\|\pi_i(x) - y\|^2 \leq \|x - y\|^2 - \|\pi_i(x) - x\|^2. \quad (1.98)$$

Из этого следует, что

$$\begin{aligned}
\|y - \psi(x)\|^2 &= \left\| y - \frac{1}{|\mathcal{J}_x|} \sum_{i \in \mathcal{J}_x} \pi_i(x) \right\|^2 = \left\| \frac{1}{|\mathcal{J}_x|} \sum_{i \in \mathcal{J}_x} (y - \pi_i(x)) \right\|^2 \leq \\
&\frac{1}{|\mathcal{J}_x|^2} \sum_{i \in \mathcal{J}_x} \|y - \pi_i(x)\|^2 \leq \\
&\leq \frac{1}{|\mathcal{J}_x|} \sum_{i \in \mathcal{J}_x} \|y - \pi_i(x)\|^2 \leq \frac{1}{|\mathcal{J}_x|} \sum_{i \in \mathcal{J}_x} \left(\|x - y\|^2 - \|\pi_i(x) - x\|^2 \right) \leq \\
&\leq \|x - y\|^2 - \frac{1}{|\mathcal{J}_x|} \sum_{i \in \mathcal{J}_x} \|\pi_i(x) - x\|^2.
\end{aligned}$$

Согласно (1.96) и (1.97), выполняется следующее неравенство:

$$\frac{1}{|\mathcal{J}_x|} \sum_{i \in \mathcal{J}_x} \|\pi_i(x) - x\|^2 > 0. \quad (1.99)$$

Следовательно,

$$\forall x \notin M, \forall y \in \mathbb{R}^n : \|\psi(x) - y\| < \|x - y\|.$$

Определение 1.10. Пусть $M \neq \emptyset$ – область допустимых значений задачи ЛП (3.1), $\psi(\cdot)$ – отображение, определяемое уравнением (1.95). Псевдопроекция $\rho_M(x)$ точки x на допустимую область многогранника M является предельной точкой последовательности $[x, \psi(x), \psi^2(x), \dots, \psi^k(x), \dots]$:

$$\lim_{k \rightarrow \infty} \|\rho_M(x) - \psi^k(x)\| = 0. \quad (1.100)$$

Правильность этого определения обеспечивается предложениями 1.8 и 1.9.

1.3 Описание метода Апекс

В этом разделе мы описываем новый итерационный метод проекционного типа для решения задачи ЛП (3.1), называемый “апекс-методом”. Апекс-метод основан на системе предиктор-корректор и состоит из двух этапов: *quest* (предиктор) и *target* (корректор). На этапе поиска вычисляется грубое начальное приближение задачи ЛП (3.1). Целевой этап

уточняет начальное приближение с заданной точностью. Основной операцией, используемой в Апекс-методе, является операция вычисления псевдопроекции в соответствии с определением 1.10. Эта операция используется как на этапе поиска, так и на целевом этапе. В следующем разделе мы опишем и исследуем параллельный алгоритм для вычисления псевдопроекции.

1.3.1 Алгоритм вычисления псевдопроекции

В этом разделе мы рассмотрим реализацию операции псевдопроекции в виде последовательных и параллельных алгоритмов. Операция псевдопроекции $\rho_M(\cdot)$ отображает произвольную точку $x \in \mathbb{R}^n$ в точку $\rho_M(x)$, принадлежащую допустимой области многогранника M , которая является областью допустимых решений задачи ЛП (3.1). Вычисление $\rho_M(x)$ организовано как итерационный процесс с использованием уравнения (1.95). Последовательная реализация этого процесса представлена алгоритмом 1.1. Дадим краткие комментарии к этой реализации. Основной итерационный процесс построения последовательности приближений Fejér's представлен циклом **repeat/until**, реализованным в шагах 4–20. На шагах 5–10 вычисляется множество \mathcal{J} индексов полупространств \hat{H}_i , нарушаемых точкой $x^{(k)}$, представляющей текущее приближение. На шагах 14–18 по уравнению (1.95) вычисляется следующее приближение $x^{(k+1)}$. Процесс завершается, когда расстояние между соседними приближениями становится меньше ε , где ε – небольшой положительный параметр. Вычислительные эксперименты (см. [70; 71]) показывают, что в случае больших задач ЛП вычисление псевдопроекции является процессом с высокой вычислительной сложностью. Поэтому мы разработали параллельную реализацию алгоритма 1.1, представленную алгоритмом 1.2.

Алгоритм 1.2 основан на модели параллельных вычислений BSF [72], разработанной для кластерной вычислительной системы. Мо-

Листинг 1.1 Вычисление псевдопроекции $\rho_M(x)$

Require: $\hat{H}_i = \{x \in \mathbb{R}^n \mid \langle a_i, x \rangle \leq b_i\}$, $M = \bigcap_{i=1}^m \hat{H}_i$, $M \neq \emptyset$

- 1: **function** $\rho_M(x)$
- 2: $k := 0$
- 3: $x^{(0)} := x$
- 4: **repeat**
- 5: $\mathcal{J} := \emptyset$
- 6: **for** $i = 1 \dots m$ **do**
- 7: **if** $\langle a_i, x^{(k)} \rangle > b_i$ **then**
- 8: $\mathcal{J} := \mathcal{J} \cup \{i\}$
- 9: **if** $\mathcal{J} = \emptyset$ **then**
- 10: **return** $x^{(k)}$
- 11: $S := 0$
- 12: **for all** $i \in \mathcal{J}$ **do**
- 13: $S := S + (\langle a_i, x^{(k)} \rangle - b_i) a_i / \|a_i\|^2$
- 14: $x^{(k+1)} := x^{(k)} - S / |\mathcal{J}|$
- 15: $k := k + 1$
- 16: **until** $\|x^{(k)} - x^{(k-1)}\| < \varepsilon$
- 17: **return** $x^{(k)}$

дель BSF использует парадигму master/worker и требует представления алгоритма в виде операций над списками с помощью функций высшего порядка *Map* и *Reduce*. В алгоритме 1.2 в качестве второго параметра функции высшего порядка *Map* используется список $\mathcal{L}_{map} = [1, \dots, m]$ порядковых номеров ограничений задачи ЛП (3.1). В качестве первого параметра функции высшего порядка *Map* мы используем параметризованную функцию

$$F_x : \mathcal{P} \rightarrow \mathbb{R}^n \times \mathbb{Z}_{\geq 0},$$

Листинг 1.2 Параллельный расчет псевдопроекции

Master	<i>l</i> th Worker (<i>l</i> =0,..., <i>L</i> -1)
1: input $n, x^{(0)}$	1: input n, m, A, b, c
2:	2: $L := \text{NumberOfWorkers}$
3: $k := 0$	3: $\mathcal{L}_{\text{map}(l)} := [lm/L, \dots, ((l+1)m/L) - 1]$
4: repeat	4: repeat
5: Bcast $x^{(k)}$	5: RecvFromMaster $x^{(k)}$
6:	6: $\mathcal{L}_{\text{reduce}(l)} := \text{Map}(F_{x^{(k)}}, \mathcal{L}_{\text{map}(l)})$
7:	7: $(u_l, \sigma_l) := \text{Reduce}(\oplus, \mathcal{L}_{\text{reduce}(l)})$
8: Gather $\mathcal{L}_{\text{reduce}}$	8: SendToMaster (u_l, σ_l)
9: $(u, \sigma) := \text{Reduce}(\oplus, \mathcal{L}_{\text{reduce}})$	9:
10: $x^{(k+1)} := u/\sigma$	10:
11: $k := k + 1$	11:
12: $\text{exit} := \ x^{(k)} - x^{(k-1)}\ < \varepsilon$	12:
13: Bcast exit	13: RecvFromMaster exit
14: until exit	14: until exit
15: output $x^{(k)}$	15:
16: stop	16: stop

определенную следующим образом:

$$F_x(i) = (u_i, \sigma_i);$$

$$u_i = \begin{cases} \pi_i(x), & \text{if } \langle a_i, x \rangle > b_i; \\ \mathbf{0}, & \text{if } \langle a_i, x \rangle \leq b_i; \end{cases} \quad (1.101)$$

$$\sigma_i = \begin{cases} 1, & \text{if } \langle a_i, x \rangle > b_i; \\ 0, & \text{if } \langle a_i, x \rangle \leq b_i. \end{cases}$$

Таким образом, функция высшего порядка $Map(F_x, \mathcal{L}_{map})$ преобразует список \mathcal{L}_{map} номеров ограничений в список пар (u_i, σ_i) . Здесь u_i – ортогональная проекция точки x на гиперплоскость H_i в случае $x \notin \hat{H}_i$, и нулевой вектор в противном случае; σ_i – показатель того, что x нарушает полупространство \hat{H}_i ($i = 1, \dots, m$):

$$Map(F_x, \mathcal{L}_{map}) = [F_x(1), \dots, F_x(m)] = [(u_1, \sigma_1), \dots, (u_m, \sigma_m)]. \quad (1.102)$$

Обозначим $\mathcal{L}_{reduce} = [(u_1, \sigma_1), \dots, (u_m, \sigma_m)]$. Определим бинарную ассоциативную операцию

$$\oplus : \mathbb{R}^n \times \mathbb{Z}_{\geq 0} \rightarrow \mathbb{R}^n \times \mathbb{Z}_{\geq 0},$$

которая является первым параметром функции высшего порядка $Reduce$, следующим образом:

$$(u', \sigma') \oplus (u'', \sigma'') = (u' + u'', \sigma' + \sigma''). \quad (1.103)$$

Функция высшего порядка $Reduce(\oplus, \mathcal{L}_{reduce})$ отображает список \mathcal{L}_{reduce} в единственную пару путем последовательного применения операции \oplus ко всем элементам списка:

$$Reduce(\oplus, \mathcal{L}_{reduce}) = (u_1, \sigma_1) \oplus \dots \oplus (u_m, \sigma_m) = (u, \sigma), \quad (1.104)$$

где

$$u = \sum_{i=1}^m u_i; \quad (1.105)$$

$$(1.106)$$

$$\sigma = \sum_{i=1}^m \sigma_i. \quad (1.107)$$

В алгоритме 1.2 параллельное выполнение работы организуется по схеме master/worker. Параллельный алгоритм включает $L + 1$ процессов: один ведущий процесс и L рабочих процессов. Ведущий процесс управляет вычислениями, распределяет работу между рабочими, собирает от

них результаты и суммирует все результаты для получения окончательного итога. Для простоты предполагается, что число ограничений m задачи ЛП (3.1) кратно числу рабочих L . На шаге 1 мастер считывает размерность пространства n и точку отсчета $x^{(0)}$. На шаге 3 мастер присваивает нулевое значение счетчику итераций k . Шаги 4–14 реализуют основной **repeat/until** цикл вычисления псевдопроекции. На Шаге 5 мастер передает текущее приближение $x^{(k)}$ всем рабочим. На Шаге 8 мастер собирает частичные результаты от всех рабочих. На Шаге 9 мастер складывает частичные результаты в пару (u, σ) , которая используется для вычисления следующего приближения $x^{(k+1)}$ на Шаге 10. На шаге 11 мастер увеличивает счетчик итераций k на 1. На шаге 12 мастер вычисляет критерий для остановки итерационного процесса и присваивает результат булевой переменной $exit$. На шаге 13 мастер передает значение булевой переменной $exit$ всем работникам. На Шаге 14 цикл **repeat/until** завершается, если булева переменная $exit$ принимает значение $true$. На шаге 15 мастер выводит последнее приближение $x^{(k)}$ как результат псевдопроекции. Шаг 16 завершает процесс мастера.

Все рабочие выполняют одни и те же программные коды, но с разными данными. На шаге 1, l -ый рабочий считывает данные задачи. На шагах 2–3 l -й рабочий определяет свой собственный подсписок $\mathcal{L}_{map(l)}$ для обработки. Для удобства мы нумеруем ограничения, начиная с нуля. Подписки разных рабочих не пересекаются, и их конкатенация представляет собой весь список, подлежащий обработке:

$$\mathcal{L}_{map} = \mathcal{L}_{map(0)} \# \dots \# \mathcal{L}_{map(L-1)}. \quad (1.108)$$

Цикл **repeat/until** рабочего l соответствует циклу **repeat/until** ведущего (шаги 4–14). На шаге 5 l -ый рабочий получает от мастера текущее приближение $x^{(k)}$. На шаге 6 l -й рабочий выполняет функцию высшего порядка Map , которая применяет параметризованную функцию $F_{x^{(k)}}$,

определенную (1.101) ко всем элементам подсписка $\mathcal{L}_{map(l)}$, в результате чего получается подсписок $\mathcal{L}_{reduce(l)}$. Шаг 7 l -го рабочего выполняет функцию высшего порядка *Reduce*, которая применяет операцию \oplus , определенную (1.103) ко всем элементам списка $\mathcal{L}_{reduce(l)}$, в результате чего получается пара (u_l, σ_l) . На шаге 8, l -ый рабочий отправляет свою результирующую пару (u_l, σ_l) ведущему. На Шаге 13, l -ый рабочий получает от мастера значение булевой переменной *exit*. Если *exit* = *true*, то рабочий процесс завершается. В противном случае цикл **repeat/until** продолжает свою работу. Операторы обмена **Bcast**, **Gather**, **RecvFromMaster** и **SendToMaster** обеспечивают синхронизацию главного и рабочего процессов.

Оценим границу масштабируемости описанного псевдопроеционного параллельного алгоритма, используя метрику стоимости модели BSF [72]. Здесь *граница масштабируемости* означает количество рабочих процессов, при котором достигается максимальное ускорение. Метрика стоимости модели BSF включает следующие параметры.

1. m – длина списка \mathcal{L}_{map} .
2. D – задержка (время, которое требуется мастеру для отправки одного байта сообщения одному рабочему).
3. t_c – время, затраченное ведущим на отправку текущего приближения $x^{(k)}$ одному рабочему и получение от него пары (u_l, σ_l) (включая задержку).
4. t_{Map} – время, затрачиваемое одним работником на обработку функции высшего порядка *Map* для всего списка \mathcal{L}_{map} .
5. t_a – время, затраченное на вычисление бинарной операции \oplus .

Согласно уравнению (14) из [72], граница масштабируемости L_{max} параллельного алгоритма может быть оценена следующим образом:

$$L_{max} = \frac{1}{2} \sqrt{\left(\frac{t_c}{t_a \ln 2}\right)^2 + \frac{t_{Map}}{t_a} + 4m} - \frac{t_c}{t_a \ln 2}. \quad (1.109)$$

Вычислим временные параметры в уравнении (2.49). Для этого введем следующие обозначения для одной итерации цикла **repeat/until**, реализованного в шагах 4–14 алгоритма 1.2:

1. c_c – количество чисел, переданных от мастера к l -му рабочему и обратно в течение одной итерации.
2. c_F – количество арифметических операций и операций сравнения, необходимых для вычисления функции F_x , заданной уравнениями (1.101).
3. c_{\oplus} – количество арифметических операций и операций сравнения, необходимых для вычисления бинарного оператора \oplus .

На шаге 5 ведущий посылает l -му рабочему один вектор размерности n . Затем, на Шаге 8, ведущий получает от l -го рабочего пару, состоящую из вектора размерности n и одного числа. Кроме того, на Шаге 13 ведущий посылает единственное булево значение l -му рабочему. Следовательно,

$$c_c = 2n + 1. \quad (1.110)$$

Принимая во внимание уравнения (1.92), (1.101), и предполагая, что $\|a_i\|^2$ вычислено заранее, получаем

$$c_F = 3n + 2. \quad (1.111)$$

Согласно (1.103), для c_{\oplus} имеет место следующее уравнение:

$$c_{\oplus} = 2n + 1. \quad (1.112)$$

Обозначим через τ_{op} время выполнения одной арифметической операции или операции сравнения, а через τ_{tr} время пересылки одного вещественного числа от одного процесса к другому (без учета задержки). Затем, используя (2.50), (1.111) и (1.112), получаем

$$t_c = c_c \tau_{tr} + 3D = (2n + 1) \tau_{tr} + 3D; \quad (1.113)$$

$$t_{Map} = c_F m \tau_{op} = (3n + 2) m \tau_{op}; \quad (1.114)$$

$$t_a = c_{\oplus} \tau_{op} = (2n + 1) \tau_{op}. \quad (1.115)$$

Подставляя правые части этих уравнений в (2.49), имеем

$$L_{max} = \frac{1}{2} \sqrt{\left(\frac{(2n+1)\tau_{tr} + 3D}{(2n+1)\tau_{op} \ln 2}\right)^2 + \left(\frac{n+1}{2n+1} + 5\right)m - \frac{(2n+1)\tau_{tr} + 3D}{(2n+1)\tau_{op} \ln 2}},$$

где n – размерность пространства, m – количество ограничений, D – задержка. Для больших значений n и m , это эквивалентно

$$L_{max} \approx O(\sqrt{m}). \quad (1.116)$$

Эта оценка показывает, что алгоритм 1.2 ограниченно масштабируемый, и масштабируемость зависит от количества ограничений m .

1.3.2 Этап поиска

Поисковый этап апекс-метода играет роль предиктора и включает следующие шаги.

1. Вычисление допустимой точки $\tilde{x} \in M$.
2. Вычисление точки вершины z .
3. Вычисление точки $u^{(0)}$, которая является псевдопроекцией вершинной точки z на допустимую область многогранника M .

Допустимую точку \tilde{x} , на шаге 1, можно вычислить по следующему уравнению:

$$\tilde{x} = \begin{cases} \mathbf{0}, & \text{if } \mathbf{0} \in M; \\ \rho_M(\mathbf{0}), & \text{if } \mathbf{0} \notin M, \end{cases} \quad (1.117)$$

где $\rho_M(\cdot)$ – операция псевдопроекции на поверхность многогранника M (см. Определение 1.10).

Шаг 2 вычисляет *точку апекса* z по следующему уравнению:

$$z = \tilde{x} + \left(\eta + \max \left\{ \frac{b_i - \langle a_i, \tilde{x}' \rangle}{\langle a_i, e_c \rangle} \mid i \in \mathcal{I}_c \right\} \right) e_c, \quad (1.118)$$

где \mathcal{I}_c – определяемое уравнением (3.40) множество индексов, для которых полупространство \hat{H}_i является c -реcessивным, а $\eta \in \mathbb{R}_{>0}$ –

положительный параметр. Следствие 1.5 гарантирует, что точка z , выбранная в соответствии с уравнением (1.118), не принадлежит ни одному c -рецессивному полупространству \hat{H}_i . Этот выбор основан на интуиции, что псевдопроекция из такой точки будет не очень далека от точного решения задачи ЛП. Интерпретация этой интуиции следует из предложения 1.6, которое гласит, что решение задачи ЛП (3.1) лежит на некоторой гиперплоскости H_i , ограничивающей c -рецессивное полупространство \hat{H}_i . Параметр η может существенно повлиять на близость точки $\rho_M(z)$ к точному решению. Оптимальное значение параметра η может быть получено путем поиска максимума целевой функции с помощью метода последовательной дихотомии.

Шаг 3 вычисляет начальное приближение $u^{(0)}$ для целевого этапа по следующему уравнению:

$$u^{(0)} = \rho_M(z). \quad (1.119)$$

Многочисленные вычислительные эксперименты показывают, что процесс вычисления псевдопроекции по Определению 1.10, начиная с внешней точки, всегда сходится к точке на границе допустимой области многогранника M . Однако на данный момент у нас нет строгого доказательства этого факта.

1.3.3 Целевой этап

Целевой этап Апекс-метода играет роль корректора и вычисляет последовательность точек

$$\{u^{(0)}, u^{(1)}, \dots, u^{(k)}, \dots\} \quad (1.120)$$

которая обладает следующими свойствами для всех $k \in \{0, 1, 2, \dots\}$:

$$u^{(k)} \in \Gamma_M; \quad (1.121)$$

$$\langle c, u^{(k)} \rangle < \langle c, u^{(k+1)} \rangle; \quad (1.122)$$

Листинг 1.3 целевой этап

Require: $\hat{H}_i = \{x \in \mathbb{R}^n \mid \langle a_i, x \rangle \leq b_i\}$, $M = \bigcap_{i=1}^m \hat{H}_i$,
 $M \neq \emptyset$

- 1: **input** $u^{(0)}$
- 2: $k := 0$
- 3: $v := u^{(k)} + \delta e_c$
- 4: $w := \rho_M(v)$
- 5: **while** $\langle c, w - u^{(k)} \rangle > \varepsilon_f$ **do**
- 6: $u := u^{(k)}$
- 7: $d := w - u^{(k)}$
- 8: **while** $\|d\| > \varepsilon_d$ **do**
- 9: **if** $(u + d) \in M$ **then**
- 10: $u := u + d$
- 11: **else**
- 12: $d := d/2$
- 13: $u^{(k+1)} := u$
- 14: $k := k + 1$
- 15: $v := u^{(k)} + \delta e_c$
- 16: $w := \rho_M(v)$
- 17: **output** $u^{(k)}$
- 18: **stop**

$$\lim_{k \rightarrow \infty} \|u^{(k)} - \bar{x}\| = 0. \quad (1.123)$$

Здесь, Γ_M обозначает множество граничных точек допустимой области многогранника M . Условие (1.121) означает, что все точки последовательности (1.120) лежат на границе многогранника M . Условие (1.122) гласит, что значение целевой функции в каждой следующей точке последовательности (1.120) больше, чем в предыдущей. Согласно условию (1.123), последовательность (1.120) сходится к точному решению задачи ЛП (3.1). Реализация целевого этапа представлена в алгоритме 1.3. Схематично работа алгоритма 1.3 показана на рисунке 1.1. Дадим краткие комментарии

по шагам алгоритма 1.3. Шаг 1 считывает начальное приближение $u^{(0)}$, построенное на этапе поиска. Шаг 2 присваивает нуль счетчику итераций k . Шаг 3 добавляет вектор δe_c к $u^{(k)}$ и присваивает результат v . Здесь, e_c – единичный вектор, параллельный c , δ – положительный параметр. Параметр δ должен быть достаточно мал, чтобы гарантировать, что $\{x \in \mathbb{R}^n \mid x = (1 - \lambda)w - \lambda u^{(k)}, 0 \leq \lambda \leq 1\} \subset \Gamma_M$. Шаг 4 вычисляет псевдопроекцию $\rho_M(v)$ и присваивает результат w . Шаги 5–19 реализуют основной цикл. Этот цикл обрабатывается, пока выполняется условие

$$\langle c, w - u^{(k)} \rangle > \varepsilon_f. \quad (1.124)$$

Здесь ε_f – небольшой положительный параметр. Шаг 6 вводит точку u , движущуюся по поверхности многогранника M от точки $u^{(k)}$ до следующего приближения $u^{(k+1)}$. На шаге 7 вычисляется вектор d , который определяет направление движения точки u . Цикл на шагах 8–14 перемещает точку u вдоль поверхности многогранника M в этом направлении настолько, насколько это возможно. Для этого вектор d последовательно делится пополам каждый раз, когда очередной шаг перемещает u за границу многогранника M . Движение прекращается, когда длина вектора d становится меньше ε_d . Здесь, ε_d – небольшой положительный параметр. Шаг 15 устанавливает следующее приближение $u^{(k+1)}$, используя значение u . Шаг 16 увеличивает счетчик итераций k на 1. Шаги 17 и 18 вычисляют новые точки v и w для следующей итерации основного цикла. Шаг 20 выводит $u^{(k)}$ как окончательное приближенное точное решение \bar{x} задачи ЛП (3.1). Следующее предложение гарантирует сходимость алгоритма 1.3.

Утверждение 1.11. Пусть допустимая область многогранника M задачи ЛП (3.1) – замкнутое ограниченное множество, и $M \neq \emptyset$. Тогда последовательность $\{u^{(k)}\}$, порожденная алгоритмом 1.3, завершается на конечном

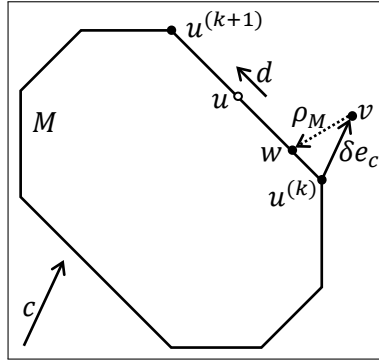


Рисунок 1.1 — На каждой итерации основного цикла алгоритм 1.3 выполняет следующие шаги. Во-первых, алгоритм строит вектор δe_c из точки $u^{(k)}$ и получает точку v . Во-вторых, псевдопроекция точки v дает точку w . Затем вычисляется вектор направления d как разность между векторами w и $u^{(k)}$. Наконец, точка u перемещается из точки $u^{(k)}$ в направлении d вдоль поверхности многогранника M на максимально возможное расстояние. В результате получаем следующее приближение $u^{(k+1)}$.

числе итераций $K \geq 0$, и

$$\langle c, u^{(0)} \rangle < \langle c, u^{(1)} \rangle < \langle c, u^{(2)} \rangle < \dots < \langle c, u^{(K)} \rangle. \quad (1.125)$$

Доказательство. Случай, когда $K = 0$, тривиален. Пусть $K > 0$ или $K = \infty$. Сначала мы покажем, что для любого $k < K$ имеет место следующее неравенство:

$$\langle c, u^{(k)} \rangle < \langle c, u^{(k+1)} \rangle. \quad (1.126)$$

Действительно, неравенство (1.124) подразумевает.

$$\langle c, u^{(k)} \rangle < \langle c, w \rangle. \quad (1.127)$$

Согласно Шагу 7 алгоритма 1.3, получается, что

$$d \neq 0. \quad (1.128)$$

Без потери общности можно предположить, что $\|w - u^{(k)}\| > \varepsilon_d$. Тогда, согласно шагам 8-15, получаем

$$u^{(k+1)} = u^{(k)} + \mu d, \quad (1.129)$$

где $\mu > 0$. С учетом неравенства (1.124), следует

$$\begin{aligned} \langle c, u^{(k+1)} \rangle &= \langle c, u^{(k)} + \mu d \rangle = \langle c, u^{(k)} + \mu (w - u^{(k)}) \rangle = \\ &= \langle c, u^{(k)} \rangle + \mu \langle c, w - u^{(k)} \rangle > \langle c, u^{(k)} \rangle. \end{aligned}$$

Теперь мы покажем, что $K < \infty$. Предположим обратное, т.е. алгоритм 1.3 порождает бесконечную последовательность точек. В этом случае мы получим монотонно возрастающую числовую последовательность

$$\langle c, u^{(0)} \rangle < \langle c, u^{(1)} \rangle < \langle c, u^{(2)} \rangle < \dots \quad (1.130)$$

Поскольку допустимая область многогранника M ограничена, последовательность (3.77) ограничена сверху. Согласно теореме о монотонной сходимости, монотонно возрастающая числовая последовательность, ограниченная сверху, сходится к своему верхнему значению. Это означает, что существует $K' \in \mathbb{N}$ такое, что

$$\forall k > K' : \langle c, u^{(k+1)} \rangle - \langle c, u^{(k)} \rangle < \varepsilon_d. \quad (1.131)$$

Из этого следует

$$\forall k > K' : \langle c, w \rangle - \langle c, u^{(k)} \rangle < \varepsilon_d \quad (1.132)$$

что эквивалентно

$$\forall k > K' : \langle c, w - u^{(k)} \rangle < \varepsilon_d. \quad (1.133)$$

Таким образом, мы получаем противоречие с критерием остановки (1.124), используемым в шаге 5 алгоритма 1.3.

Понятие псевдопроекции является обобщением понятия *метрической проекции*, которое можно определить следующим образом [66].

Определение 1.12. Пусть Q – замкнутое выпуклое множество в \mathbb{R}^n , и $Q \neq \emptyset$. Метрическая проекция $P_Q(x)$ точки $x \in \mathbb{R}^n$ на множество Q определяется уравнением

$$P_Q(x) = \arg \min \{ \|x - q\| \mid q \in Q \}. \quad (1.134)$$

Очевидно, что предложение 1.11 остается верным, если псевдопроекцию $\rho_M(\cdot)$ заменить метрической проекцией $P_M(\cdot)$ в алгоритме 1.3. В этом случае имеет место следующее более сильное предложение.

Утверждение 1.13. Пусть в алгоритме 1.3 псевдопроекция $\rho_M(\cdot)$ заменена метрической проекцией $P_M(\cdot)$. Тогда алгоритм 1.3 сходится к точному решению \bar{x} задачи ЛП (3.1).

Доказательство. Пусть \bar{u} – конечная точка последовательности $\{u^{(k)}\}$, порожденной алгоритмом 1.3. Предположим обратное, т.е. $\bar{u} \neq \bar{x}$. Это эквивалентно

$$\langle c, \bar{u} \rangle < \langle c, \bar{x} \rangle. \quad (1.135)$$

Пусть $S_\delta(v)$ обозначает открытый n -шар радиуса δ с центром v , где

$$v = \bar{u} + \delta e_c. \quad (1.136)$$

Согласно (1.135), следует, что

$$S_\delta(v) \cap M \neq \emptyset. \quad (1.137)$$

Пусть

$$w = \arg \min \{ \|x - v\| \mid x \in S_\delta(v) \cap M \}, \quad (1.138)$$

т.е.,

$$w = P_M(v). \quad (1.139)$$

Легко убедиться в справедливости следующего неравенства:

$$\langle c, w \rangle > \langle c, \bar{u} \rangle. \quad (1.140)$$

Условия (3.90), (1.139) и (1.140) устанавливают, что \bar{u} не является конечной точкой последовательности $\{u^{(k)}\}$, порожденной алгоритмом 1.3. Таким образом, мы получаем противоречие.

В случае использования псевдопроекции в алгоритме 1.3, сходимость к точному решению основана на интуитивном предположении, что $\rho_M(v) \rightarrow P_M(v)$ с $\delta \rightarrow 0$. Однако строгое доказательство этого факта выходит за рамки данной статьи.

Таблица 2 — Спецификации вычислительного кластера «Tornado SUSU»

Параметр	Значение
Количество процессорных узлов	480
Процессор	Intel Xeon X5680 (6 ядер, 3,33 ГГц)
Процессоры на узел	2
Память на узел	24 ГБ DDR3
Шина данных	InfiniBand QDR (40 Гбит/с)
Операционная система	Linux CentOS

1.4 Внедрение и вычислительные эксперименты

Мы реализовали параллельную версию Апекс-метода на C++, используя BSF-каркас [73], который основан на модели параллельных вычислений BSF [72]. BSF--каркас инкапсулирует все аспекты, связанные с распараллеливанием программы с помощью библиотеки MPI. Исходный код этой реализации находится в свободном доступе на сайте <https://github.com/leonid-sokolinsky/Apex-method>. Используя эту программу, мы исследовали масштабируемость Апекс-метода. Вычислительные эксперименты проводились на вычислительном кластере «Торнадо ЮУрГУ» [74], характеристики которого приведены в таблице 2. В качестве тестовых задач мы использовали случайные синтетические задачи ЛП, сгенерированные программой FRaGenLP [75]. Проверка решений, полученных апекс-методом, проводилась с помощью программы VaLiPro [76]. Результаты вычислительных экспериментов представлены на рисунке 1.3, демонстрирующем зависимость ускорения от количества используемых процессорных узлов. Здесь ускорение α определяется как отношение времени $T(1)$, требующегося параллельному алгоритму, использующему один главный узел и один рабочий узел для решения задачи, ко времени $T(L)$, требующегося параллельному алгоритму, использующему один главный узел и L рабочих узлов для решения той же задачи:

$$\alpha = \frac{T(1)}{T(L)}. \quad (1.141)$$

Вычисления проводились со следующими размерностями: 5 000, 7 500 и 10 000. Количество неравенств составляло 10 002, 15 002 и 20 002, соответственно. Эксперименты показали, что граница масштабируемости параллельного алгоритма Апекс-метода существенно зависит от размерности задачи ЛП. При $n = 5\,000$ граница масштабируемости составила примерно 55 процессорных узлов. Для задачи размерностью $n = 7\,500$ эта граница увеличилась до 80 узлов, а для задачи размерностью $n = 10\,000$ она приблизилась к 100 узлам. Дальнейшее увеличение размерности задачи приводило к ошибке компилятора: “недостаточно памяти”. Следует отметить, что вычисления проводились в формате двойной точности с плавающей точкой, занимающем 64 бита в памяти компьютера. Попытка использовать формат с плавающей точкой одинарной точности, занимающий 32 бита, не удалась, так как алгоритм апекс-метода перестал сходиться.

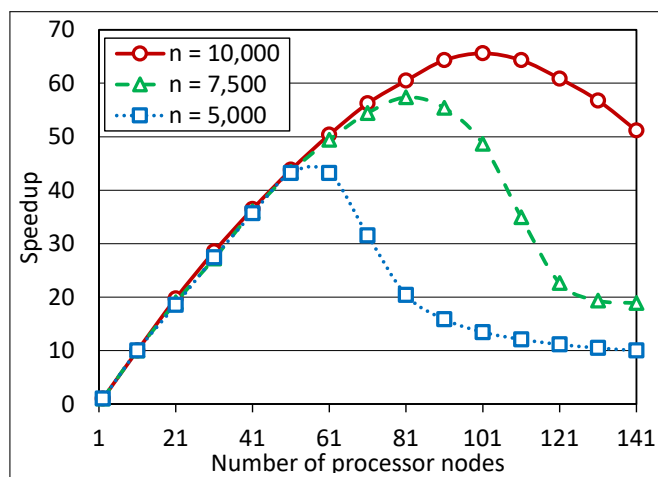


Рисунок 1.3 — Кривые демонстрируют зависимость ускорения от количества процессорных узлов, используемых параллельным алгоритмом Апекс-метода. Ускорение - это отношение времени, требуемого параллельному алгоритму, использующему один главный узел и один рабочий узел для решения задачи, к времени, требуемого параллельному алгоритму, использующему один главный узел и L рабочих узлов для решения той же задачи. Эксперименты показывают, что масштабируемость параллельного алгоритма Апекс-метода существенно зависит от размерности n задачи ЛП

2. ВИЗУАЛЬНОЕ ПРЕДСТАВЛЕНИЕ n -МЕРНЫХ МНОГОГРАННИКОВ

2.1 Математическая модель визуального представления задачи ЛП

Рассмотрим задачу ЛП в следующей форме:

$$\bar{x} = \arg \max \{ \langle c, x \rangle \mid Ax \leq b, x \in \mathbb{R}^n \}, \quad (2.1)$$

где $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$ и $c \neq 0$. Здесь $\langle \cdot, \cdot \rangle$ обозначает скалярное произведение двух векторов. Мы предполагаем, что ограничение $x \geq 0$ также включено в систему $Ax \leq b$ в виде неравенств

$$\begin{aligned} -x_1 + 0 + \dots + 0 &\leq 0; \\ 0 - x_2 + 0 + \dots + 0 &\leq 0; \\ \dots &\dots \\ 0 + \dots + 0 - x_n &\leq 0. \end{aligned}$$

Вектор c является градиентом линейной целевой функции

$$f(x) = c_1x_1 + \dots + c_nx_n. \quad (2.2)$$

Обозначим через M допустимую область задачи (2.1):

$$M = \{x \in \mathbb{R}^n \mid Ax \leq b\}. \quad (2.3)$$

Везде далее мы предполагаем, что M является непустым ограниченным множеством. Это означает, что M представляет собой выпуклый замкнутый многогранник в пространстве \mathbb{R}^n , и множество решений задачи (2.1) не является пустым.

Пусть $\tilde{a}_i \in \mathbb{R}^n$ — вектор, образованный элементами i -той строки матрицы A . Тогда матричное неравенство $Ax \leq b$ можно представить в виде системы неравенств

$$\langle \tilde{a}_i, x \rangle \leq b_i, i = 1, \dots, m. \quad (2.4)$$

Везде далее мы будем предполагать, что для всех $i = 1, \dots, m$

$$\tilde{a}_i \neq 0. \quad (2.5)$$

Обозначим через H_i гиперплоскость, задаваемую уравнением

$$\langle \tilde{a}_i, x \rangle = b_i \quad (1 \leq i \leq m). \quad (2.6)$$

Таким образом,

$$H_i = \{x \in \mathbb{R}^n \mid \langle \tilde{a}_i, x \rangle = b_i\}. \quad (2.7)$$

Определение 2.1. Под полупространством H_i^+ , порождаемым гиперплоскостью H_i , будем понимать полупространство, определяемое формулой

$$H_i^+ = \{x \in \mathbb{R}^n \mid \langle \tilde{a}_i, x \rangle \leq b_i\}. \quad (2.8)$$

Везде далее мы будем предполагать, что задача (2.1) является невырожденной, то есть

$$\forall i \neq j : H_i \neq H_j \quad (i, j \in \{1, \dots, m\}). \quad (2.9)$$

Определение 2.2. Полупространство H_i^+ , порождаемое гиперплоскостью H_i , является *рецессивным* относительно вектора c , если

$$\forall x \in H_i, \forall \lambda \in \mathbb{R}_{>0} : x - \lambda c \in H_i^+ \wedge x - \lambda c \notin H_i. \quad (2.10)$$

Другими словами, луч, исходящий из гиперплоскости H_i в направлении противоположном вектору c , полностью лежит в H_i^+ , но не в H_i .

Утверждение 2.3. Необходимым и достаточным условием рецессивности полупространства H_i^+ относительно вектора c является условие

$$\langle \tilde{a}_i, c \rangle > 0. \quad (2.11)$$

Доказательство. Докажем сначала необходимость. Пусть выполняется условие (2.10). Из (2.7) следует

$$x = \frac{b_i \tilde{a}_i}{\|\tilde{a}_i\|^2} \in H_i. \quad (2.12)$$

Из (2.5) следует

$$\lambda = \frac{1}{\|\tilde{a}_i\|^2} \in \mathbb{R}_{>0}. \quad (2.13)$$

Сопоставляя (2.10),(2.12) и (2.13) получаем

$$\begin{aligned} \frac{b_i \tilde{a}_i}{\|\tilde{a}_i\|^2} - \frac{1}{\|\tilde{a}_i\|^2} c &\in H_i^+; \\ \frac{b_i \tilde{a}_i}{\|\tilde{a}_i\|^2} - \frac{1}{\|\tilde{a}_i\|^2} c &\notin H_i. \end{aligned}$$

С учетом (2.7), (2.8) отсюда следует, что

$$\left\langle \tilde{a}_i, \frac{b_i \tilde{a}_i}{\|\tilde{a}_i\|^2} - \frac{1}{\|\tilde{a}_i\|^2} c \right\rangle < b_i. \quad (2.14)$$

Выполнив несложные преобразования неравенства (2.14), получаем (2.11).

Доказательство достаточности проведем от противного. Предположим, что имеет место (2.11) и существуют $x \in H_i$ и $\lambda > 0$ такие, что

$$x - \lambda c \notin H_i^+ \vee x - \lambda c \in H_i.$$

В соответствии с (2.7), (2.8) это означает

$$\langle \tilde{a}_i, x - \lambda c \rangle \geq b_i,$$

что равносильно

$$\langle \tilde{a}_i, x \rangle - \lambda \langle \tilde{a}_i, c \rangle \geq b_i.$$

Так как $\lambda > 0$, то в силу (2.11) отсюда получаем

$$\langle \tilde{a}_i, x \rangle > b_i,$$

что противоречит предположению $x \in H_i$.

Определение 2.4. Зафиксируем точку $z \in \mathbb{R}^n$ такую, что полупространство

$$H_c^+ = \{x \in \mathbb{R}^n \mid \langle c, x - z \rangle \leq 0\} \quad (2.15)$$

включает в себя многогранник M :

$$M \subset H_c^+.$$

Полупространство H_c^+ в этом случае будем называть *целевым полупространством*, а гиперплоскость H_c , определяемую уравнением

$$H_c = \{x \in \mathbb{R}^n \mid \langle c, x - z \rangle = 0\}, \quad (2.16)$$

будем называть *целевой гиперплоскостью*.

Обозначим через $\pi_c(x)$ ортогональную проекцию точки x на целевую гиперплоскость H_c :

$$\pi_c(x) = x - \frac{\langle c, x - z \rangle}{\|c\|^2} c. \quad (2.17)$$

Здесь $\|\cdot\|$ обозначает евклидову норму. Определим *расстояние* $\rho_c(x)$ от точки $x \in H_c^+$ до целевой гиперплоскости H_c следующим образом:

$$\rho_c(x) = \|\pi_c(x) - x\|. \quad (2.18)$$

Сопоставляя (2.15), (2.17) и (2.18) находим, что в этом случае расстояние $\rho_c(x)$ может быть вычислено следующим образом:

$$\rho_c(x) = \frac{\langle c, z - x \rangle}{\|c\|}. \quad (2.19)$$

Справедливо следующее предложение.

Утверждение 2.5. Для любых $x, y \in H_c^+$

$$\rho_c(x) \leq \rho_c(y) \Leftrightarrow \langle c, x \rangle \geq \langle c, y \rangle.$$

Доказательство. Используя (2.19), получаем

$$\begin{aligned} \rho_c(x) \leq \rho_c(y) &\Leftrightarrow \frac{\langle c, z - x \rangle}{\|c\|} \leq \frac{\langle c, z - y \rangle}{\|c\|} \\ &\Leftrightarrow \langle c, z - x \rangle \leq \langle c, z - y \rangle \\ &\Leftrightarrow \langle c, z \rangle + \langle c, -x \rangle \leq \langle c, z \rangle + \langle c, -y \rangle \\ &\Leftrightarrow \langle c, -x \rangle \leq \langle c, -y \rangle \\ &\Leftrightarrow \langle c, x \rangle \geq \langle c, y \rangle. \end{aligned}$$

В соответствии с предложением 2.5 задача (2.1) эквивалентна следующей задаче:

$$\bar{x} = \arg \min \{ \rho_c(x) | x \in M \}. \quad (2.20)$$

Определение 2.6. Пусть полупространство H_i^+ является рецессивным относительно вектора c . *Целевой проекцией* $\gamma_i(x)$ точки $x \in \mathbb{R}^n$ на рецессивное полупространство H_i^+ называется точка, определяемая формулой

$$\gamma_i(x) = x - \sigma_i(x)c, \quad (2.21)$$

где

$$\sigma_i(x) = \min \{ \sigma \in \mathbb{R}_{\geq 0} \mid x - \sigma c \in H_i^+ \}.$$

Примеры целевых проекций в \mathbb{R}^2 приведены на рис. 2.1.

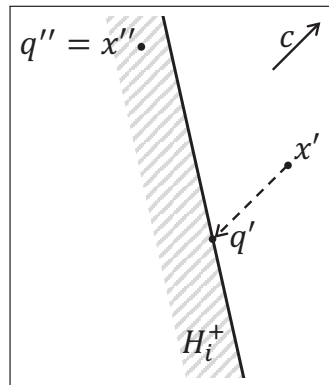


Рисунок 2.1 — Целевые проекции в пространстве \mathbb{R}^2 : $\gamma_i(x') = q'$; $\gamma_i(x'') = q'' = x''$.

Следующее предложение предоставляет формулу для вычисления целевой проекции на полупространство, рецессивное относительно вектора c .

Утверждение 2.7. Пусть полупространство H_i^+ , определяемое неравенством

$$\langle \tilde{a}_i, x \rangle \leq b_i, \quad (2.22)$$

является рецессивным относительно вектора c . Пусть

$$g \notin H_i^+. \quad (2.23)$$

Тогда

$$\gamma_i(g) = g - \frac{\langle \tilde{a}_i, g \rangle - b_i}{\langle \tilde{a}_i, c \rangle} c. \quad (2.24)$$

Доказательство. В соответствии с определением 2.6 имеем

$$\gamma_i(g) = g - \sigma_i(g)c,$$

где

$$\sigma_i(x) = \min \{ \sigma \in \mathbb{R}_{\geq 0} \mid x - \sigma c \in H_i^+ \}.$$

Таким образом, нам необходимо показать, что

$$\frac{\langle \tilde{a}_i, g \rangle - b_i}{\langle \tilde{a}_i, c \rangle} = \min \{ \sigma \in \mathbb{R}_{\geq 0} \mid x - \sigma c \in H_i^+ \}. \quad (2.25)$$

Рассмотрим прямую L , задаваемую параметрическим уравнением

$$L = \{ g + \tau c \mid \tau \in \mathbb{R} \}.$$

Пусть точка q является пересечением прямой L с гиперплоскостью H_i :

$$q = L \cap H_i. \quad (2.26)$$

Тогда q должна удовлетворять уравнению

$$q = g + \tau' c \quad (2.27)$$

при некотором $\tau' \in \mathbb{R}$. Подставим правую часть уравнения (2.27) в уравнение (3.5) вместо x :

$$\langle \tilde{a}_i, g + \tau' c \rangle = b_i.$$

Отсюда

$$\begin{aligned} \langle \tilde{a}_i, g \rangle + \tau' \langle \tilde{a}_i, c \rangle &= b_i, \\ \tau' &= \frac{b_i - \langle \tilde{a}_i, g \rangle}{\langle \tilde{a}_i, c \rangle}. \end{aligned} \quad (2.28)$$

Подставив вместо τ' правую часть уравнения (2.28) в формулу (2.27) получаем

$$q = g + \frac{b_i - \langle \tilde{a}_i, g \rangle}{\langle \tilde{a}_i, c \rangle} c,$$

что эквивалентно

$$q = g - \frac{\langle \tilde{a}_i, g \rangle - b_i}{\langle \tilde{a}_i, c \rangle} c. \quad (2.29)$$

Так как $q \in H_i$ в соответствии с (2.26), формула (3.22) будет иметь место, если мы покажем, что

$$\forall \sigma \in \mathbb{R}_{>0} : \sigma < \frac{\langle \tilde{a}_i, g \rangle - b_i}{\langle \tilde{a}_i, c \rangle} \Rightarrow g - \sigma c \notin H_i^+. \quad (2.30)$$

Предположим противное, то есть существует $\sigma' > 0$ такое, что

$$\sigma' < \frac{\langle \tilde{a}_i, g \rangle - b_i}{\langle \tilde{a}_i, c \rangle} \quad (2.31)$$

и

$$g - \sigma' c \in H_i^+. \quad (2.32)$$

Тогда из (2.22) и (2.32) следует

$$\langle \tilde{a}_i, g - \sigma' c \rangle \leq b_i,$$

что равносильно

$$\langle \tilde{a}_i, g \rangle - b_i \leq \sigma' \langle \tilde{a}_i, c \rangle. \quad (2.33)$$

В силу предложения 2.3 имеем $\langle \tilde{a}_i, c \rangle > 0$. Поэтому (2.33) равносильно

$$\frac{\langle \tilde{a}_i, g \rangle - b_i}{\langle \tilde{a}_i, c \rangle} \leq \sigma'.$$

Получили противоречие с (2.31).

Определение 2.8. Пусть $g \in H_c$. Целевой проекцией $\gamma_M(g)$ точки g на многогранник M называется точка, определяемая формулой

$$\gamma_M(g) = g - \sigma_M(g)c, \quad (2.34)$$

где

$$\sigma_M(g) = \min \{ \sigma \in \mathbb{R}_{\geq 0} \mid g - \sigma c \in M \}.$$

Если

$$\neg \exists \sigma \in \mathbb{R}_{\geq 0} : g - \sigma c \in M,$$

полагаем $\gamma_M(g) = \vec{\infty}$ — точка, бесконечно удаленная от M .

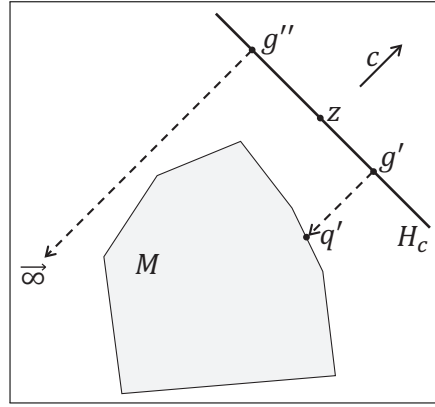


Рисунок 2.2 — Целевые проекции на многогранник M в пространстве \mathbb{R}^2 :
 $\gamma_M(g') = q'$; $\gamma_M(g'') = \infty$.

Примеры целевых проекций на многогранник M в \mathbb{R}^2 приведены на рис. 2.2.

Определение 2.9. *Рецептивным полем* $\mathfrak{G}(z, \eta, \delta) \subset H_c$ плотности $\delta \in \mathbb{R}_{>0}$ с центром в точке $z \in H_c$ и рангом $\eta \in \mathbb{N}$ будем называть конечное упорядоченное множество точек, удовлетворяющих следующим условиям:

$$z \in \mathfrak{G}(z, \eta, \delta); \quad (2.35)$$

$$\forall g \in \mathfrak{G}(z, \eta, \delta) : \|g - z\| \leq \eta \delta \sqrt{n}; \quad (2.36)$$

$$\forall g', g'' \in \mathfrak{G}(z, \eta, \delta) : g' \neq g'' \Rightarrow \|g' - g''\| \geq \delta; \quad (2.37)$$

$$\forall g' \in \mathfrak{G}(z, \eta, \delta) \exists g'' \in \mathfrak{G}(z, \eta, \delta) : \|g' - g''\| = \delta; \quad (2.38)$$

$$\forall x \in \text{Co}(\mathfrak{G}(z, \eta, \delta)) \exists g \in \mathfrak{G}(z, \eta, \delta) : \|g - x\| \leq \frac{1}{2} \delta \sqrt{n}. \quad (2.39)$$

Здесь $\text{Co}(X)$ обозначает выпуклую оболочку конечного множества точек $X = \{x^{(1)}, \dots, x^{(K)}\} \subset \mathbb{R}^n$:

$$\text{Co}(X) = \left\{ \sum_{i=1}^K \lambda_i x^{(i)} \mid \lambda_i \in \mathbb{R}_{\geq 0}, \sum_{i=1}^K \lambda_i = 1 \right\}.$$

Точки рецептивного поля будем называть *рецептивными точками*.

Формула (2.35) в определении 2.9 означает, что точка, находящаяся в центре рецептивного поля, принадлежит этому полю. Из формулы (2.36) следует, что любая точка g рецептивного поля отстоит от центральной точки z на расстояние не более $\eta \delta \sqrt{n}$. В соответствии с формулой (2.37) для

любых двух различных точек $g' \neq g''$ рецептивного поля расстояние между ними не может быть меньше δ . Формула (2.38) говорит, что для любой точки g' рецептивного поля найдется точка g'' , принадлежащая этому же полю, такая, что расстояние между g' и g'' будет равно δ . Формула (2.39) означает, что для любой точки x , принадлежащей выпуклой оболочке рецептивного поля, в этом поле найдется точка g , отстоящая от x на расстояние не более $\frac{1}{2}\delta\sqrt{n}$. Пример рецептивного поля в пространстве \mathbb{R}^3 приведен на рис. 2.3.

Опишем конструктивный метод построения рецептивного поля. Без ограничения общности мы можем предполагать, что $c_n \neq 0$. Построим в \mathbb{R}^n следующий ортогональный базис, включающий в себя вектор c :

$$\begin{aligned}
c^{(0)} &= c = (c_1, c_2, c_3, c_4, \dots, c_{n-1}, c_n); \\
c^{(1)} &= \begin{cases} \left(-\frac{1}{c_1} \sum_{i=2}^n c_i^2, c_2, c_3, c_4, \dots, c_{n-1}, c_n\right), & \text{если } c_1 \neq 0; \\ (1, 0, \dots, 0), & \text{если } c_1 = 0; \end{cases} \\
c^{(2)} &= \begin{cases} \left(0, -\frac{1}{c_2} \sum_{i=3}^n c_i^2, c_3, c_4, \dots, c_{n-1}, c_n\right), & \text{если } c_2 \neq 0; \\ (0, 1, 0, \dots, 0), & \text{если } c_2 = 0; \end{cases} \\
c^{(3)} &= \begin{cases} \left(0, 0, -\frac{1}{c_3} \sum_{i=4}^n c_i^2, c_4, \dots, c_{n-1}, c_n\right), & \text{если } c_3 \neq 0; \\ (0, 0, 1, 0, \dots, 0), & \text{если } c_3 = 0; \end{cases} \\
&\dots\dots\dots \\
c^{(n-2)} &= \begin{cases} \left(0, \dots, 0, -\frac{1}{c_{n-2}} \sum_{i=n-1}^n c_i^2, c_{n-1}, c_n\right), & \text{если } c_{n-2} \neq 0; \\ (0, \dots, 0, 1, 0, 0), & \text{если } c_{n-2} = 0; \end{cases} \\
c^{(n-1)} &= \begin{cases} \left(0, \dots, 0, -\frac{c_n^2}{c_{n-1}}, c_n\right), & \text{если } c_{n-1} \neq 0; \\ (0, \dots, 0, 0, 1, 0), & \text{если } c_{n-1} = 0. \end{cases}
\end{aligned}$$

Непосредственно проверяется, что

$$\forall i, j \in \{0, 1, \dots, n-1\}, i \neq j : \langle c^{(i)}, c^{(j)} \rangle = 0.$$

В частности

$$\forall i = 1, \dots, n-1 : \langle c, c^{(i)} \rangle = 0. \tag{2.40}$$

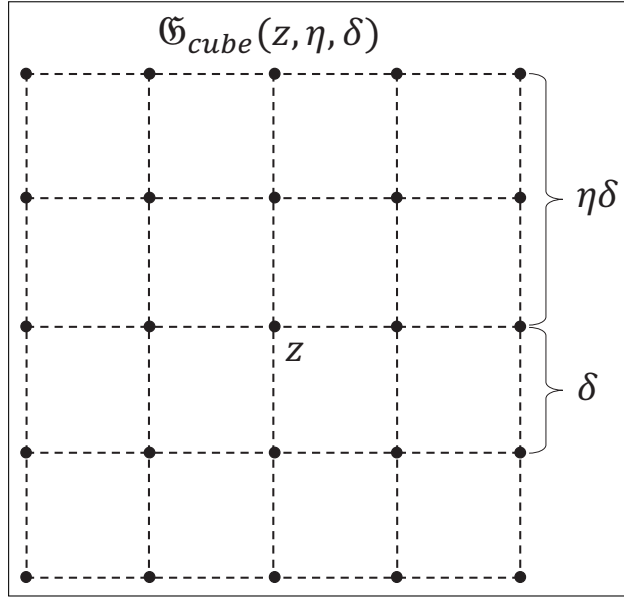


Рисунок 2.3 — Рецептливное поле в пространстве \mathbb{R}^3 .

Следующее предложение показывает, что линейное подпространство размерности $(n - 1)$, порождаемое ортогональными векторами c_1, \dots, c_{n-1} , является гиперплоскостью, параллельной гиперплоскости H_c .

Утверждение 2.10. Определим линейное подпространство $S_c \subset \mathbb{R}^n$ размерности $n - 1$:

$$S_c = \left\{ \sum_{i=1}^{n-1} \lambda_i c^{(i)} \mid \lambda_i \in \mathbb{R} \right\}. \quad (2.41)$$

Тогда

$$\forall s \in S_c : s + z \in H_c. \quad (2.42)$$

Доказательство. Пусть $s \in S_c$, то есть

$$s = \lambda_1 c^{(1)} + \dots + \lambda_{n-1} c^{(n-1)}.$$

Тогда

$$\langle c, (s + z) - z \rangle = \lambda_1 \langle c, c^{(1)} \rangle + \dots + \lambda_{n-1} \langle c, c^{(n-1)} \rangle.$$

Отсюда в силу (2.40) получаем

$$\langle c, (s + z) - z \rangle = 0.$$

Сопоставляя это с (3.29), имеем $s + z \in H_c$.

Листинг 2.1 Построение рецептивного поля $\mathfrak{G}(z, \eta, \delta)$

Require: $z \in H_c, \eta \in \mathbb{N}, \delta \in \mathbb{R}_{>0}$

```

1:  $\mathfrak{G} := \emptyset$ 
2: for  $i_{n-1} = 0 \dots 2\eta$  do
3:    $s_{n-1} := i_{n-1}\delta - \eta\delta$ 
4:   for  $i_{n-2} = 0 \dots 2\eta$  do
5:      $s_{n-2} := i_{n-2}\delta - \eta\delta$ 
6:     ...
7:   for  $i_1 = 0 \dots 2\eta$  do
8:      $s_1 := i_1\delta - \eta\delta$ 
9:      $s := \mathbf{0}$ 
10:    for  $j = 1 \dots n - 1$  do
11:       $s := s + s_j e^{(j)}$ 
12:     $\mathfrak{G} := \mathfrak{G} \cup \{s + z\}$ 

```

Определим векторы

$$e^{(i)} = \frac{c^{(i)}}{\|c^{(i)}\|} \quad (i = 1, \dots, n - 1), \quad (2.43)$$

образующие ортонормированный базис подпространства S_c .

Процедура построения рецептивного поля представлена в виде алгоритма 3.2. Этот алгоритм строит рецептивное поле $\mathfrak{G}(z, \eta, \delta)$, состоящее из

$$K_{\mathfrak{G}} = (2\eta + 1)^{n-1} \quad (2.44)$$

точек, расположенных в узлах регулярной решетки, имеющей форму гиперквадрата (гиперкуба размерности $n - 1$) с длиной ребра, равной $2\eta\delta$. Длина ребра ячейки гиперквадрата равна δ . В соответствии с шагом 13 алгоритма 3.2 и предложением 2.10 этот гиперквадрат лежит в гиперплоскости H_c и имеет центр в точке z . Недостаток алгоритма 3.2 состоит в том, что количество вложенных циклов **for** зависит от размерности пространства. Эту проблему можно решить с помощью функции G , вычисляющей

Листинг 2.2 Функция G вычисляет точку рецептивного поля по ее номеру k

Require: $z \in H_c, \eta \in \mathbb{N}, \delta \in \mathbb{R}_{>0}$

```
1: function  $G(k, n, z, \eta, \delta)$ 
2:   for  $j = (n - 1) \dots 1$  do
3:      $l_j := \lfloor k / (2\eta + 1)^{j-1} \rfloor$ 
4:      $k := k \bmod (2\eta + 1)^{j-1}$ 
5:    $g := z$ 
6:   for  $j = 1 \dots (n - 1)$  do
7:      $g := g + (l_j \delta - \eta \delta) e^{(j)}$ 
8:    $G := g$ 
```

точку рецептивного множества по ее порядковому номеру (нумерация начинается с нуля; порядок определяется алгоритмом 3.2). Реализация функции G представлена в виде алгоритма 2.2. Следующее предложение 2.11 дает оценку временной сложности алгоритма 2.2.

Утверждение 2.11. Алгоритм 2.2 допускает реализацию с временной сложностью

$$c_G = 4n^2 + 5n - 9, \quad (2.45)$$

где n — размерность пространства.

Под временной сложностью здесь понимается количество арифметических операций и операций сравнения, необходимых для выполнения алгоритма.

Доказательство. Рассмотрим низкоуровневую реализацию алгоритма 2.2, представленную в виде алгоритма 2.3.

Листинг 2.3 Низкоуровневая реализация алгоритма 2.2

```
1:  $p := 2\eta + 1; r := \eta\delta; h := p^{n-2}; g := z$ 
2:  $j := n - 1$ 
3: repeat
4:    $l_j := \lfloor k/h \rfloor$ 
5:    $k := k \bmod h$ 
6:    $h := h/p$ 
7:    $j := j - 1$ 
8: until  $j = 0$ 
9:  $j := 1$ 
10: repeat
11:    $w_j := l_j\delta - r$ 
12:    $i := 1$ 
13:   repeat
14:      $g_i := g_i + w_j e_i^{(j)}$ 
15:      $i := i + 1$ 
16:   until  $i > n$ 
17:    $j := j + 1$ 
18: until  $j = n$ 
```

Значения, вычисляемые на шагах 1–2 алгоритма 2.3, не зависят от номера рецептивной точки k и поэтому могут считаться константами. Цикл **repeat/until** на шагах 3–8 выполняется $(n - 1)$ раз и требует $c_{3:8} = 5(n - 1)$ операций. Вложенный цикл **repeat/until** на шагах 13–16 выполняется n раз и требует $c_{13:16} = 4n$ операций. Внешний цикл **repeat/until** на шагах 10–18 выполняется $(n - 1)$ раз и требует $c_{10:18} = (4 + c_{13-16})(n - 1) = 4(n^2 - 1)$ операций. В сумме получаем

$$c_G = c_{3:8} + c_{10:18} = 4n^2 + 5n - 9.$$

Утверждение доказано.

Утверждение 2.12. Временная сложность алгоритма 2.2 может быть оценена как $O(n^2)$.

Определение 2.13. Пусть $z \in H_c$. Зафиксируем $\eta \in \mathbb{N}$, $\delta \in \mathbb{R}_{>0}$. Образом $\mathfrak{I}(z, \eta, \delta)$, порожденным рецептивным полем $\mathfrak{G}(z, \eta, \delta)$, будем называть упорядоченное множество вещественных чисел

$$\mathfrak{I}(z, \eta, \delta) = \{\rho_c(\gamma_M(g)) \mid g \in \mathfrak{G}(z, \eta, \delta)\}. \quad (2.46)$$

Порядок чисел в образе определяется порядком соответствующих точек рецептивного поля.

Функцию построения образа $\mathfrak{I}(z, \eta, \delta)$ в виде списка чисел представлена в виде алгоритма 2.4. Здесь $[]$ обозначает пустой список, а $\#$ обозначает операцию конкатенации списков.

Листинг 2.4 Построение образа $\mathfrak{I}(z, \eta, \delta)$

Require: $z \in H_c, \eta \in \mathbb{N}, \delta \in \mathbb{R}_{>0}$

```

1: function  $\mathfrak{I}(z, \eta, \delta)$ 
2:    $\mathfrak{I} := []$ 
3:   for  $k = 0 \dots ((2\eta + 1)^{n-1} - 1)$  do
4:      $g_k := G(k, n, z, \eta, \delta)$ 
5:      $\mathfrak{I} := \mathfrak{I} \# [\rho_c(\gamma_M(g_k))]$ 

```

Покажем как полученный образ может быть использован для решения задачи ЛП. Пусть $\langle \tilde{a}_i, c \rangle > 0$. Это означает, что полупространство H_i^+ является рецессивным относительно вектора c (см. предложение 2.3). Пусть имеется точка $u \in H_i \cap M$. Допустим, что нам удалось создать искусственную нейронную сеть DNN, получающую на входе образ $\mathfrak{I}(\pi_c(u), \eta, \delta)$ окрестности точки u , и выдающую на выходе точку u' такую, что

$$u' = \arg \min \{\rho_c(x) \mid x \in H_i \cap M\}.$$

Тогда мы можем построить алгоритм 2.5, решающий задачу линейного программирования (2.20) с помощью DNN.

Листинг 2.5 Линейное программирование с использованием DNN

Require: $u^{(1)} \in H_i \cap M$, $\langle \tilde{a}_i, c \rangle > 0$, $z \in H_c$; $\eta \in \mathbb{N}$, $\delta \in \mathbb{R}_{>0}$

- 1: $k := 1$
- 2: **repeat**
- 3: $\mathcal{I} := \mathfrak{I}(u^{(k)}, \eta, \delta)$
- 4: $u^{(k+1)} := \text{DNN}(\mathcal{I})$
- 5: $k := k + 1$
- 6: **until** $u^{(k)} \neq u^{(k-1)}$
- 7: $\bar{x} := u^{(k)}$
- 8: **stop**

2.2 Параллельный алгоритм построения образа задачи ЛП

При решении задач ЛП большой размерности и с большим количеством ограничений алгоритм 2.4 построения образа задачи ЛП может потребовать значительных временных затрат. В этом разделе приводится его параллельная версия, позволяющая существенно сократить время решения задачи ЛП с помощью алгоритма 2.5. Параллельная версия алгоритма 2.4 строится на основе модели параллельных вычислений BSF [72; 77], ориентированной на кластерные вычислительные системы. Модель BSF использует парадигму мастер–рабочие и требует представление алгоритма в форме операций над списками с использованием функций высшего порядка *Map* и *Reduce*, определенных в формализме Бёрда–Миртенса (Bird–Meertens formalism) [78]. Модель BSF также предоставляет метрику для аналитической оценки масштабируемости параллельного алгоритма, удовлетворяющего указанным требованиям.

Представим алгоритм 2.4 в форме операций над списками с использованием функций высшего порядка *Map* и *Reduce*. В качестве списка, обрабатываемого функцией высшего порядка *Map*, возьмем список номеров неравенств системы (2.4):

$$\mathcal{L}_{map} = [1, \dots, m]. \quad (2.47)$$

Обозначим $\mathbb{R}_\infty = \mathbb{R} \cup \{\infty\}$. Определим следующим образом параметризованную функцию $F_k : \{1, \dots, m\} \rightarrow \mathbb{R}_\infty$, являющуюся первым параметром функции высшего порядка *Map*:

$$F_k(i) = \begin{cases} \rho_c(\gamma_i(g_k)), & \text{если } \langle \tilde{a}_i, c \rangle > 0 \text{ и } \gamma_i(g_k) \in M; \\ \infty, & \text{если } \langle \tilde{a}_i, c \rangle \leq 0 \text{ или } \gamma_i(g_k) \notin M, \end{cases} \quad (2.48)$$

где $g_k = G(k, n, z, \eta, \delta)$ вычисляется с помощью алгоритма 2.2, а $\gamma_i(g_k)$ вычисляется по формуле (3.26). С неформальной точки зрения функция F_k отображает номер полупространства H_i^+ в расстояние от целевой проекции до целевой гиперплоскости, если H_i^+ является рецессивным относительно c (см. предложение 2.3) и целевая проекция принадлежит M . В противном случае F_k возвращает специальное значение ∞ .

Функция высшего порядка *Map* преобразует список \mathcal{L}_{map} в список \mathcal{L}_{reduce} путем применения функции F_k к каждому элементу списка \mathcal{L}_{map} :

$$\mathcal{L}_{reduce} = \text{Map}(F_k, \mathcal{L}_{map}) = [F_k(1), \dots, F_k(m)] = [\rho_1, \dots, \rho_m].$$

Определим бинарную ассоциативную операцию $\ominus : \mathbb{R}_\infty \rightarrow \mathbb{R}_\infty$ следующим образом:

$$\begin{aligned} \infty \ominus \infty &= \infty; \\ \forall \alpha \in \mathbb{R} : \alpha \ominus \infty &= \alpha; \\ \forall \alpha, \beta \in \mathbb{R} : \alpha \ominus \beta &= \min(\alpha, \beta). \end{aligned}$$

С неформальной точки зрения операция \ominus вычисляет минимальное из двух чисел.

Листинг 2.6 Построение образа \mathcal{J} с использованием *Map* и *Reduce*

Require: $z \in H_c, \eta \in \mathbb{N}, \delta \in \mathbb{R}_{>0}$

```
1: input  $n, m, A, b, c, z, \eta, \delta$ 
2:  $\mathcal{J} := []$ 
3:  $\mathcal{L}_{map} := [1, \dots, m]$ 
4: for  $k = 0 \dots ((2\eta + 1)^{n-1} - 1)$  do
5:    $\mathcal{L}_{reduce} := \text{Map}(F_k, \mathcal{L}_{map})$ 
6:    $\rho := \text{Reduce}(\oplus, \mathcal{L}_{reduce})$ 
7:    $\mathcal{J} := \mathcal{J} \# [\rho]$ 
8: output  $\mathcal{J}$ 
9: stop
```

Функция высшего порядка *Reduce* преобразует список \mathcal{L}_{reduce} в атомарное значение $\rho \in \mathbb{R}_\infty$ путем последовательного применения операции \oplus ко всему списку:

$$\text{Reduce}(\oplus, \mathcal{L}_{reduce}) = \rho_1 \oplus \rho_2 \oplus \dots \oplus \rho_m = \rho.$$

Построение образа \mathcal{J} задачи ЛП с использованием функций высшего порядка *Map* и *Reduce* представлено в виде алгоритма 2.6. Параллельная версия алгоритма 2.6 строится на основе алгоритмического шаблона 2 из [72]. Результат представлен в виде алгоритма 2.7. Прокомментируем параллельный алгоритм 2.7. Для простоты мы будем предполагать, что количество ограничений m кратно количеству рабочих L и нумерация неравенств начинается с нуля. Параллельный алгоритм включает в себя $L + 1$ процесс: один процесс-мастер (кратко — мастер) и L процессов-рабочих (кратко — рабочие). Мастер управляет вычислениями. Первоначально в качестве образа \mathcal{J} берется пустой список (шаг 2 мастера). Текущий номер k полагается равным нулю (шаг 3 мастера). На шагах 4–15 мастер организует цикл **repeat/until**, в котором строится образ \mathcal{J} задачи ЛП. На шаге 5 мастер посылает номер очередной рецептивной точки k

Листинг 2.7 Параллельный алгоритм построение образа \mathfrak{J} задачи ЛП

Мастер	Рабочий ($l=0, \dots, L-1$)
1: input n	1: input $n, m, A, b, c, z, \eta, \delta$
2: $\mathfrak{J} := []$	2: $L := \text{NumberOfWorkers}$
3: $k := 0$	3: $\mathcal{L}_{\text{map}(l)} := [lm/L, \dots, ((l+1)m/L) - 1]$
4: repeat	4: repeat
5: SendToWorkers k	5: RecvFromMaster k
6:	6: $\mathcal{L}_{\text{reduce}(l)} := \text{Map}(\mathbb{F}_k, \mathcal{L}_{\text{map}(l)})$
7:	7: $\rho_l := \text{Reduce}(\oplus, \mathcal{L}_{\text{reduce}(l)})$
8: RecvFromWorkers $[\rho_0, \dots, \rho_{L-1}]$	8: SendToMaster ρ_l
9: $\rho := \text{Reduce}(\oplus, [\rho_0, \dots, \rho_{L-1}])$	9:
10: $\mathfrak{J} := \mathfrak{J} \# [\rho]$	10:
11: $k := k + 1$	11:
12: $\text{exit} := (k \geq (2\eta + 1)^{n-1})$	12:
13: SendToWorkers exit	13: RecvFromMaster exit
14: until exit	14: until exit
15: output \mathfrak{J}	15:
16: stop	16: stop

всем рабочим. На шаге 8 мастер ожидает получение частичных результатов от всех рабочих. Эти результаты редуцируются в одно значение, которое добавляется в образ \mathfrak{J} (шаги 9–10 мастера). Шаг 11 увеличивает счетчик итераций k на единицу. На шаге 12 мастер вычисляет критерий завершения в виде логического выражения $(k \geq (2\eta + 1)^{n-1})$, значение которого присваивается булевой переменной exit . В соответствии с формулой (2.44) значение true будет означать, что точки рецептного поля закончились. На шаге 13 мастер посылает значение булевой переменной exit всем рабочим. Если обработаны не все точки рецептивного поля, то на шаге 14 происходит переход к выполнению следующей итерации цикла **repeat/until**. В

противном случае мастер выводит построенный образ \mathfrak{J} (шаг 15) и завершает свою работу (шаг 16).

Каждый l -тый рабочий выполняет общую последовательность действий, но над своей частью $\mathcal{L}_{map(l)}$ списка \mathcal{L}_{map} , которая определяется на шаге 3. На шаге 4 рабочий входит в цикл **repeat/until**. На шаге 5 он получает номер очередной точки k , принадлежащей рецептивному полю. На шаге 6 рабочий обрабатывает свой подсписок $\mathcal{L}_{map(l)}$, используя функцию высшего порядка Map , которая для каждого элемента подсписка выполняет параметризованную функцию F_k , определенную с помощью формулы (2.48). В результате получается подсписок $\mathcal{L}_{reduce(l)}$, содержащий расстояния $F_k(i)$ от целевой гиперплоскости H_c до целевых проекций рецептивной точки g_k на гиперплоскости H_i для всех i из подсписка $\mathcal{L}_{map(l)}$. На шаге 7 рабочий с помощью функции высшего порядка $Reduce$ редуцирует подсписок $\mathcal{L}_{reduce(l)}$ в атомарное значение ρ_l , используя ассоциативную бинарную операцию \oplus , вычисляющую минимальное расстояние. Полученный частный результат пересылается мастеру (шаг 8 рабочего). На шаге 13 рабочий ожидает получение от мастера значения булевой переменной $exit$. Если получено значение `false`, рабочий продолжает выполнение цикла **repeat/until** (шаг 14 рабочего). В противном случае процесс рабочего завершается на шаге 16.

Дадим *аналитическую оценку границы масштабируемости* параллельного алгоритма 2.7 с использованием стоимостной метрики модели параллельных вычислений BSF [72]. Под границей масштабируемости здесь понимается число рабочих, на котором достигается максимум ускорения. Стоимостная метрика модели BSF включает в себя следующие стоимостные параметры для цикла **repeat/until** (шаги 4–14) параллельного алгоритма 2.7:

1. m – длина списка \mathcal{L}_{map} .
2. D – латентность (время пересылки одного байта от мастера к рабочему).

3. t_c – время, затрачиваемое мастером на пересылку одному рабочему координат рецептивной точки и получения от него расстояния от этой точки до целевой проекции (включая латентность).
4. t_{Map} – время, затрачиваемое одним рабочим на выполнение функции высшего порядка Map для всего списка \mathcal{L}_{map} .
5. t_a – время, затрачиваемое на выполнение одной бинарной операции \Downarrow

В соответствии с формулой (14) из [72] граница масштабируемости параллельного алгоритма 2.7 может быть оценена следующим образом:

$$L_{max} = \frac{1}{2} \sqrt{\left(\frac{t_c}{t_a \ln 2}\right)^2 + \frac{t_{Map}}{t_a} + 4m} - \frac{t_c}{t_a \ln 2}. \quad (2.49)$$

Вычислим оценку для временных параметров формулы (2.49). Для этого введем следующие обозначения в рамках одной итерации цикла **repeat/until** (шаги 4–14) параллельного алгоритма 2.7:

1. c_c – количество чисел, пересылаемых от мастера к рабочему и обратно в рамках одной итерации.
2. c_{Map} – количество арифметических операций и операций сравнения, выполняемых на шаге 5 последовательного алгоритма 2.6.
3. c_a – количество операций, необходимых для выполнения бинарной операции \Downarrow

В начале каждой итерации мастер посылает каждому рабочему номер рецептивной точки. В ответ рабочий посылает расстояние от этой точки до целевой проекции. Следовательно

$$c_c = 2. \quad (2.50)$$

В контексте алгоритма 2.6

$$c_{Map} = (c_G + c_{F_k}) m, \quad (2.51)$$

где c_G — количество операций, необходимых для вычисления координат точки g_k , c_{F_k} — количество операций, необходимых для вычисления функции $F_k(i)$ по формуле (2.48) в предположении, что координаты точки g_k

уже вычислены. Оценка значения c_G дана в предложении 2.11. Оценим значение c_{F_k} . В соответствии с (3.26) вычисление целевой проекции $\gamma_i(g_k)$ требует $(6n - 2)$ арифметических операций. Из (2.19) следует, что вычисление $\rho_c(x)$ составляет $(5n - 1)$ арифметических операций. Проверка условия $x \in M$ в соответствии с (2.4) потребует $m(2n - 1)$ арифметических операций и m операций сравнения. Таким образом

$$c_{F_k} = 2mn + 11n - 3. \quad (2.52)$$

Подставив в (2.51) правые части формул (2.45) и (2.52), получаем

$$c_{Map} = 4n^2m + 2m^2n + 16nm - 12m. \quad (2.53)$$

Для выполнения бинарной операции \oplus необходимо выполнить одну операцию сравнения:

$$c_a = 1. \quad (2.54)$$

Пусть τ_{op} — среднее время выполнения арифметических операций и операций сравнения, τ_{tr} — среднее время для пересылки одного вещественного числа без учета латентности. Тогда, используя формулы (2.50), (2.53) и (2.54), получаем

$$t_c = c_c \tau_{tr} + 2D = 2(\tau_{tr} + D); \quad (2.55)$$

$$t_{Map} = c_{Map} \tau_{op} = (4n^2m + 2m^2n + 16nm - 12m) \tau_{op}; \quad (2.56)$$

$$t_a = c_a \tau_{op} = \tau_{op}. \quad (2.57)$$

Подставив в (2.49) правые части формул (2.55) – (2.57), получим следующую оценку границы масштабируемости параллельного алгоритма 2.7:

$$L_{max} = \frac{1}{2} \sqrt{\left(\frac{2(\tau_{tr} + D)}{\tau_{op} \ln 2} \right)^2 + 4n^2m + 2m^2n + 16nm - 12m} - \frac{2(\tau_{tr} + D)}{\tau_{op} \ln 2}.$$

Для больших значений m и n это эквивалентно

$$L_{max} \approx O(\sqrt{2n^2m + m^2n + 8nm - 6m}). \quad (2.58)$$

Если предположить, что $m = O(n)$, то из (2.58) следует

$$L_{max} \approx O(n\sqrt{n}), \quad (2.59)$$

где n — размерность пространства. Оценка (2.59) позволяет сделать вывод, что параллельный алгоритм 2.7 демонстрирует превосходную масштабируемость.

Пусть $L_{max} = O(n^\alpha)$. Алгоритм демонстрирует *превосходную масштабируемость*, если $\alpha > 1$; алгоритм демонстрирует *хорошую масштабируемость*, если $\alpha = 1$; алгоритм демонстрирует *ограниченную масштабируемость*, если $0 < \alpha < 1$; алгоритм *не масштабируется*, если $\alpha = 0$.

В следующем разделе мы проверим аналитическую оценку (2.59) путем проведения масштабных вычислительных экспериментов на реальной кластерной вычислительной системе.

2.3 Вычислительные эксперименты

Нами была выполнена параллельная реализация алгоритма 2.7 в виде программы ViLiPP (Visualization of Linear Programming Problem) на языке C++ с использованием программного BSF-каркаса [73; 79]. BSF-каркас базируется на модели параллельных вычислений BSF и инкапсулирует все аспекты, связанные с распараллеливанием программы с использованием библиотеки MPI [80] и программного интерфейса OpenMP [81]. Исходные коды программы ViLiPP свободно доступны в сети Интернет по адресу <https://github.com/nikolay-olkhovskiy/LP-visualization-MPI>. С использованием параллельной программы ViLiPP мы провели эксперименты по исследованию масштабируемости алгоритма 2.7 на кластерной вычислительной системе «Торнадо ЮУрГУ» [82], характеристики которой приведены в таблице 4.

Таблица 4 — Характеристики кластера «Торнадо ЮУрГУ»

Параметр	Значение
Количество процессорных узлов	480
Процессоры	Intel Xeon X5680 (6 ядер, 3.33 GHz)
Количество процессоров в узле	2
Оперативная память узла	24 GB DDR3
Соединительная сеть	InfiniBand QDR (40 Gbit/s)
Операционная система	Linux CentOS

С помощью генератора задач «FRaGenLP» [83; 84] для проведения вычислительных экспериментов были сгенерированы три случайные задачи ЛП, параметры которых приведены в таблице ???. Количество ненулевых значений матрицы A задачи (2.1) всех случаях составило 100%. Для всех задач ранг рецептивного поля η полагался равным 2. В соответствии с формулой (2.44) мощность рецептивного поля демонстрировала экспоненциальный рост с увеличением размерности пространства.

Таблица 5 — Параметры тестовых задач ЛП

Идентификатор задачи	Число переменных	Число ограничений	Процент ненулевых значений в A	Мощность рецептивного поля
$LP7$	7	4016	100%	15 625
$LP6$	6	4014	100%	3 125
$LP5$	5	4012	100%	625

Результаты вычислительных экспериментов приведены в таблице 6 и на рис. 2.7. Во всех запусках каждому рабочему выделялся отдельный процессорный узел. Еще один дополнительный процессорный узел выделялся для работы мастера. Вычислительные эксперименты показали, что с ростом размерности задачи наблюдается рост границы масштабируемости программы ViLiPP: для $LP5$ максимум кривой ускорения достигается в районе 190 узлов, для $LP6$ максимум располагается в районе 260 узлов, а для $LP7$ он приблизительно равен 326 узлам. При этом наблюдается экспоненциальный рост времени решения задачи: образ задачи $LP5$ на 11 процессорных узлах строится за 10 сек., а построение образа задачи $LP7$

Таблица 6 — Время построения образа задач ЛП (сек.)

Число процессорных узлов	LP5	LP6	LP7
11	9.81	54.45	303.78
56	1.93	10.02	59.43
101	1.55	6.29	33.82
146	1.39	4.84	24.73
191	1.35	4.20	21.10
236	1.38	3.98	19.20
281	1.45	3.98	18.47
326	1.55	4.14	18.30

на таком же количестве узлов требует уже 5 мин. Дополнительный вычислительный эксперимент показал, что построение образа задачи при $n = 9$ на 11 процессорных узлах занимает 1.5 часа.

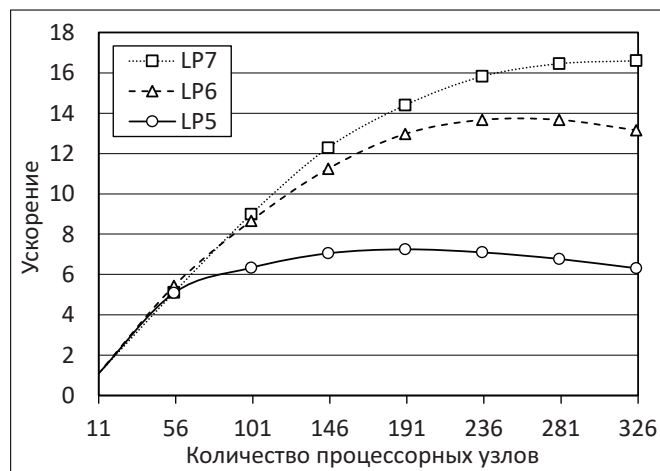


Рисунок 2.7 — Графики ускорения параллельной программы ViLiPP для задач ЛП различного размера

Проведенные эксперименты позволяют сделать вывод, что при современном уровне развития вычислительной техники применение искусственных нейронных сетей для решения задач ЛП на основе предложенного метода визуализации может быть эффективным для задач размерности, не превышающей 100, с количеством ограничений до 100 000.

3. МЕТОД ПОВЕРХНОСТНОГО ДВИЖЕНИЯ

3.1 Теоретический базис

В этом разделе описывается теоретический фундамент, на котором базируется метод поверхностного движения.

Сформулируем задачу ЛП в следующем виде:

$$\bar{x} = \arg \max_{x \in \mathbb{R}^n} \{ \langle c, x \rangle \mid Ax \leq b \}. \quad (3.1)$$

где $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$, $m > 1$, $c \neq \mathbf{0}$. Здесь $\langle \cdot, \cdot \rangle$ обозначает скалярное произведение двух векторов. Мы предполагаем, что ограничение $x \geq \mathbf{0}$ также включено в матричное неравенство $Ax \leq b$ в форме

$$-x \leq \mathbf{0}.$$

Обозначим через \mathcal{P} множество индексов, нумерующих строки матрицы A :

$$\mathcal{P} = \{1, \dots, m\}. \quad (3.2)$$

Линейная целевая функция задачи (3.1) имеет вид

$$f(x) = \langle c, x \rangle. \quad (3.3)$$

Вектор c в данном случае является градиентом целевой функции $f(x)$.

Пусть $a_i \in \mathbb{R}^n$ обозначает вектор, представляющий i -тую строку матрицы A . Мы предполагаем, что $a_i \neq \mathbf{0}$ для всех $i \in \mathcal{P}$. Обозначим через \hat{H}_i замкнутое полупространство, определяемое неравенством $\langle a_i, x \rangle \leq b_i$, а через H_i — ограничивающую его гиперплоскость:

$$\hat{H}_i = \{x \in \mathbb{R}^n \mid \langle a_i, x \rangle \leq b_i\}; \quad (3.4)$$

$$H_i = \{x \in \mathbb{R}^n \mid \langle a_i, x \rangle = b_i\}. \quad (3.5)$$

Определим допустимый многогранник

$$M = \bigcap_{i \in \mathcal{P}} \hat{H}_i, \quad (3.6)$$

представляющий множество допустимых точек задачи ЛП (3.1). Заметим, что M в этом случае будет замкнутым выпуклым множеством. Мы будем предполагать, что $M \neq \emptyset$, то есть задача ЛП (3.1) имеет решение. Обозначим через $\Gamma(M)$ множество граничных точек многогранника M . Под граничной точкой множества $M \subset \mathbb{R}^n$ понимается точка в \mathbb{R}^n , для которой любая открытая ее окрестность в \mathbb{R}^n имеет непустое пересечение как с множеством M , так и с его дополнением.

Утверждение 3.1. Пусть M — допустимый многогранник задачи ЛП (3.1), определяемый формулой (3.6). Тогда для любой точки $u \in \Gamma(M)$ существует $\varepsilon > 0$ такой, что для любой граничной точки w , принадлежащей ε -окрестности $\mathcal{B}_\varepsilon(u)$ точки u , найдется $i' \in \mathcal{P}$, для которого справедливо $u, w \in H_{i'}$:

$$\forall u \in \Gamma(M) \exists \varepsilon > 0 : \forall w \in \mathcal{B}_\varepsilon(u) \cap \Gamma(M) \exists i' \in \mathcal{P} : u, w \in H_{i'}. \quad (3.7)$$

Доказательство. Обозначим

$$\mathcal{P}_u = \{i \in \mathcal{P} \mid u \in H_i\}, \quad (3.8)$$

то есть \mathcal{P}_u — множество индексов всех гиперплоскостей H_i , которым принадлежит точка u . Положим

$$\mathcal{P}_{\setminus u} = \mathcal{P} \setminus \mathcal{P}_u, \quad (3.9)$$

то есть $\mathcal{P}_{\setminus u}$ — множество индексов всех гиперплоскостей H_i , которым точка u не принадлежит. Определим

$$\delta = \min \{ \text{dist}(u, H_i) \mid i \in \mathcal{P}_{\setminus u} \}, \quad (3.10)$$

где $\text{dist}(u, H_i)$ обозначает евклидово расстояние от точки u до гиперплоскости H_i . В данном случае $\text{dist}(u, H_i) = \frac{\langle a_i, u \rangle - b_i}{\|a_i\|}$. Очевидно

$$\delta > 0. \quad (3.11)$$

Возьмем ε , удовлетворяющий условию

$$0 < \varepsilon < \delta. \quad (3.12)$$

Тогда для любого $w \in \mathcal{B}_\varepsilon(u) \cap \Gamma(M)$ имеем

$$\forall i \in \mathcal{P}_{\setminus u} : w \notin H_i. \quad (3.13)$$

Поскольку w — граничная точка, то отсюда следует, что найдется $i' \in \mathcal{P}_u$ такой, что

$$w \in H_{i'}. \quad (3.14)$$

Заметим, что в силу (3.8) также имеем

$$u \in H_{i'}. \quad (3.15)$$

Утверждение доказано.

Определение 3.2. Целевой проекцией точки $z \in \mathbb{R}^n$ на гиперплоскость H_i называется точка $\gamma_i(z) \in \mathbb{R}^n \cup \{\infty\}$, определяемая формулой

$$\gamma_i(z) = \begin{cases} L(z) \cap H_i, & \text{если } \langle a_i, c \rangle \neq 0; \\ \infty, & \text{если } \langle a_i, c \rangle = 0, \end{cases} \quad (3.16)$$

где $L(z)$ — прямая, проходящую через точку z параллельно вектору c :

$$L(z) = \{y \in \mathbb{R}^n \mid y = z + \lambda c, \lambda \in \mathbb{R}\}. \quad (3.17)$$

Другими словами, если вектор c не параллелен гиперплоскости H_i , то целевой проекцией точки z на гиперплоскость H_i является точка пересечения этой гиперплоскости с прямой, проходящей через точку z параллельно вектору c (см. рис. 3.1). В случае, когда вектор c параллелен гиперплоскости H_i , целевая проекция полагается равной бесконечно удаленной точке ∞ .

Утверждение 3.3. Пусть $\langle a_i, c \rangle \neq 0$. Тогда

$$\gamma_i(z) = z - \frac{\langle a_i, z \rangle - b_i}{\langle a_i, c \rangle} c. \quad (3.18)$$

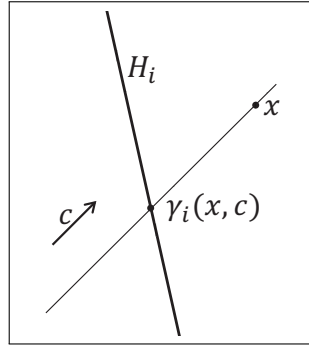


Рисунок 3.1 — Целевая проекция $\gamma_i(z)$ точки z на гиперплоскость H_i .

Доказательство. В соответствии с (3.17) и (3.18) имеем

$$\gamma_i(z) = z + \lambda c \quad (3.19)$$

при некотором $\lambda \in \mathbb{R}$. С другой стороны, в соответствии с (3.5) имеем

$$\langle a_i, \gamma_i(z) \rangle = b_i. \quad (3.20)$$

Подставим правую часть формулы (3.19) в формулу (3.20) вместо $\gamma_i(z)$:

$$\langle a_i, z + \lambda c \rangle = b_i. \quad (3.21)$$

Отсюда

$$\lambda = -\frac{\langle a_i, z \rangle - b_i}{\langle a_i, c \rangle}. \quad (3.22)$$

Подставив правую часть формулы (3.22) вместо λ в формулу (3.19), получаем

$$\gamma_i(z) = z - \frac{\langle a_i, z \rangle - b_i}{\langle a_i, c \rangle} c. \quad (3.23)$$

Утверждение доказано.

Определение 3.4. Целевым смещением точки $z \in \mathbb{R}^n$ относительно гиперплоскости H_i называется скалярная величина $\beta_i(z)$, вычисляемая по формуле

$$\beta_i(z) = -\frac{\langle a_i, z \rangle - b_i}{\langle a_i, c \rangle} \|c\|. \quad (3.24)$$

Далее мы для краткости будем использовать термин ”смещение подразумеваемая по этим ”целевое смещение”. Обозначим

$$e_c = \frac{c}{\|c\|}. \quad (3.25)$$

Тогда формулу (3.18) можно переписать в виде

$$\gamma_i(z) = z + \beta_i(z)e_c, \quad (3.26)$$

что равносильно

$$\beta_i(z)e_c = \gamma_i(z) - z. \quad (3.27)$$

С учетом (3.25) отсюда следует

$$|\beta_i(z)| = \|\gamma_i(z) - z\|. \quad (3.28)$$

Таким образом, $|\beta_i(z)|$ является расстоянием от точки z до ее целевой проекции на гиперплоскость H_i .

Определение 3.5. Целевой гиперплоскостью $H_c(z)$, проходящей через точку z , будем называть гиперплоскость, задаваемую формулой

$$H_c(z) = \{x \in \mathbb{R}^n \mid \langle c, x \rangle = \langle c, z \rangle\}. \quad (3.29)$$

Справедливо следующее утверждение.

Утверждение 3.6. Зафиксируем произвольную точку $z \in \mathbb{R}^n$. Тогда для любых точек $z', z'' \in H_c(z)$, $z' \neq z''$ справедливо

$$\langle c, \gamma_i(z') \rangle < \langle c, \gamma_i(z'') \rangle \Leftrightarrow \beta_i(z') < \beta_i(z'') \quad (3.30)$$

для всех $i \in \mathcal{P}$.

Доказательство. Поскольку $z', z'' \in H_c(z)$, а c является нормалью к гиперплоскости $H_c(z)$, справедливы следующие два равенства

$$\langle c, z' - z \rangle = 0; \quad (3.31)$$

$$\langle c, z'' - z \rangle = 0. \quad (3.32)$$

Следовательно

$$\langle c, z'' \rangle = \langle c, z \rangle = \langle c, z' \rangle. \quad (3.33)$$

Используя (3.26) и (3.25), отсюда получаем следующую цепочку эквивалентных неравенств.

$$\begin{aligned}
\langle c, \gamma_i(z') \rangle < \langle c, \gamma_i(z'') \rangle &\Leftrightarrow \langle c, z' + \beta_i(z')e_c \rangle < \langle c, z'' + \beta_i(z'')e_c \rangle \\
&\Leftrightarrow \langle c, z' \rangle + \langle c, \beta_i(z')e_c \rangle < \langle c, z'' \rangle + \langle c, \beta_i(z'')e_c \rangle \\
&\Leftrightarrow \langle c, \beta_i(z')e_c \rangle < \langle c, \beta_i(z'')e_c \rangle \\
&\Leftrightarrow \langle c, \beta_i(z')c / \|c\| \rangle < \langle c, \beta_i(z'')c / \|c\| \rangle \\
&\Leftrightarrow \frac{\beta_i(z')}{\|c\|} \langle c, c \rangle < \frac{\beta_i(z'')}{\|c\|} \langle c, c \rangle \\
&\Leftrightarrow \beta_i(z') < \beta_i(z'').
\end{aligned}$$

Утверждение доказано.

Следуя [85] дадим определение рецессивного полупространства.

Определение 3.7. Полупространство \hat{H}_i называется рецессивным, если

$$\forall x \in H_i, \forall \lambda > 0 : x + \lambda c \notin \hat{H}_i. \quad (3.34)$$

Геометрический смысл этого определения состоит в том, что луч, исходящий в направлении вектора c из любой точки гиперплоскости, ограничивающей рецессивное полупространство, не имеет общих точек с этим полупространством, за исключением начальной. Известно [85], что следующее условие является необходимым и достаточным для того, чтобы полупространство \hat{H}_i было рецессивным:

$$\langle a_i, c \rangle > 0. \quad (3.35)$$

Рецессивное полупространство обладает следующими свойствами.

Утверждение 3.8. Пусть полупространство \hat{H}_i является рецессивным. Тогда любая прямая, параллельная вектору c , пересекает гиперплоскость H_i в единственной точке.

Данное свойство непосредственно вытекает из того факта, что гиперплоскость H_i , ограничивающая рецессивное полупространство \hat{H}_i по определению не может быть параллельна вектору c .

Утверждение 3.9. Пусть полупространство \hat{H}_i является рецессивным. Тогда

$$x \in \hat{H}_i \Leftrightarrow \beta_i(x) \geq 0. \quad (3.36)$$

Действительно, в силу (3.24) имеем

$$\beta_i(z) = -\frac{\langle a_i, z \rangle - b_i}{\langle a_i, c \rangle} \|c\|. \quad (3.37)$$

Из (3.4) следует

$$\langle a_i, z \rangle - b_i \leq 0. \quad (3.38)$$

Принимая во внимание (3.35), отсюда получаем

$$x \in \hat{H}_i \Leftrightarrow \beta_i(z) \geq 0. \quad (3.39)$$

Определим

$$\mathcal{I} = \{i \in \mathcal{P} \mid \langle a_i, c \rangle > 0\}, \quad (3.40)$$

то есть \mathcal{I} представляет множество индексов, для которых полупространство \hat{H}_i является рецессивным. Поскольку допустимый многогранник M представляет собой ограниченное множество, имеем

$$\mathcal{I} \neq \emptyset. \quad (3.41)$$

Положим

$$\hat{M} = \bigcap_{i \in \mathcal{I}} \hat{H}_i. \quad (3.42)$$

Очевидно, что \hat{M} является выпуклым, замкнутым, неограниченным многогранником. Будем называть его рецессивным. Из (3.6) и (3.40) следует

$$M \subset \hat{M}. \quad (3.43)$$

Обозначим через $\Gamma(\hat{M})$ множество граничных точек рецессивного многогранника \hat{M} . Согласно утверждению 3 в [85] имеем

$$\bar{x} \in \Gamma(\hat{M}), \quad (3.44)$$

то есть решение задачи ЛП (3.1) лежит на границе рецессивного многогранника \hat{M} .

Утверждение 3.10. Пусть \hat{M} — рецессивный многогранник, определяемый формулой (3.42). Тогда для любой точки $u \in \Gamma(\hat{M})$ существует $\varepsilon > 0$ такой, что для любой граничной точки w , принадлежащей ε -окрестности $\mathcal{B}_\varepsilon(u)$ точки u , найдется $i' \in \mathcal{I}$, для которого справедливо $u, w \in H_{i'}$:

$$\forall u \in \Gamma(\hat{M}) \exists \varepsilon > 0 : \forall w \in \mathcal{B}_\varepsilon(u) \cap \Gamma(\hat{M}) \exists i' \in \mathcal{I} : u, w \in H_{i'}. \quad (3.45)$$

Доказательство. Доказательство идентично доказательству утверждения 3.1.

Определение 3.11. Целевой проекцией точки $z \in \mathbb{R}^n$ на границу $\Gamma(\hat{M})$ рецессивного многогранника \hat{M} называется точка $\hat{\gamma}(z)$, вычисляемая по формуле

$$\hat{\gamma}(z) = L(z) \cap \Gamma(\hat{M}), \quad (3.46)$$

где $L(z)$ — прямая, проходящую через точку z параллельно вектору c :

$$L(z) = \{y \in \mathbb{R}^n \mid y = z + \lambda c, \lambda \in \mathbb{R}\}. \quad (3.47)$$

Скалярную величину $\beta(z) \in \mathbb{R}$, удовлетворяющую уравнению

$$\hat{\gamma}(z) = z + \beta(z)c \quad (3.48)$$

будем называть целевым смещением (или кратко — смещением) точки z относительно границы рецессивного многогранника \hat{M} .

Заметим, что корректность этого определения базируется на свойстве 3.8.

Следующее утверждение предоставляет формулу для вычисления целевой проекции на границу рецессивного многогранника.

Утверждение 3.12. Пусть задана произвольная точка $z \in \mathbb{R}^n$. Положим

$$i' = \arg \min \{\beta_i(z) \mid i \in \mathcal{I}\}. \quad (3.49)$$

Тогда

$$\hat{\gamma}(z) = \gamma_{i'}(z). \quad (3.50)$$

Другими словами, целевая проекция точки z на границу рецессивного многогранника \hat{M} совпадает с проекцией этой точки на гиперплоскость $H_{i'}$, имеющей минимальное смещение относительно z .

Доказательство. Зафиксируем произвольную точку $z \in \mathbb{R}^n$. В соответствии с определением 3.11 построим прямую, параллельную вектору c , которая проходит через точку z :

$$L = \{y \in \mathbb{R}^n \mid y = z + \lambda c, \lambda \in \mathbb{R}\}. \quad (3.51)$$

Имеем

$$\hat{\gamma}(z) = L \cap \Gamma(\hat{M}). \quad (3.52)$$

Согласно свойству 3.8 для любого $i \in \mathcal{I}$ прямая L пересекает H_i в точности в одной точке. Обозначим

$$Y = \bigcup_{i \in \mathcal{I}} \{y_i\}, \quad (3.53)$$

где $y_i = L \cap H_i$. То есть Y — множество точек, в которых прямая L пересекает границы рецессивных полупространств. По определению 3.2 имеем

$$\gamma_i(z) = y_i; \quad (3.54)$$

$$y_i \in H_i \quad (3.55)$$

для всех $i \in \mathcal{I}$. В силу (3.42) также имеем

$$\hat{\gamma}(z) \in Y. \quad (3.56)$$

Положим

$$i' = \arg \min \{\beta_i(z) \mid i \in \mathcal{I}\}. \quad (3.57)$$

Выберем произвольный $i'' \in \mathcal{I}$ такой, что

$$\beta_{i''} > \beta_{i'} \quad (3.58)$$

(если такого не существует, то доказывать нечего). В силу (3.26) и (3.54)

имеем

$$y' = z + \beta_{i'} e_c; \quad (3.59)$$

$$y'' = z + \beta_{i''} e_c. \quad (3.60)$$

Отсюда

$$y'' = y' + (\beta_{i''} - \beta_{i'}) e_c, \quad (3.61)$$

что в силу (3.58) и (3.25) равносильно

$$y'' = y' + \frac{|\beta_{i''} - \beta_{i'}|}{\|c\|} c. \quad (3.62)$$

В соответствии с (3.34) и (3.55), из (3.62) следует

$$y'' \notin \hat{H}_{i'}. \quad (3.63)$$

Это означает, что

$$y'' \notin \hat{M}. \quad (3.64)$$

Следовательно

$$\hat{\gamma}(z) = y' = \gamma_{i'}(z), \quad (3.65)$$

где

$$i' = \arg \min \{ \beta_i(z) \mid i \in \mathcal{I} \}. \quad (3.66)$$

Утверждение доказано.

Следующее утверждение предоставляет формулу для вычисления смещения точки относительно границы рецессивного многогранника.

Утверждение 3.13. Пусть задана произвольная точка $z \in \mathbb{R}^n$. Тогда

$$\beta(z) = \frac{\langle c, \hat{\gamma}(z) - z \rangle}{\|c\|^2}. \quad (3.67)$$

Доказательство. В соответствии с (3.48) точки $\hat{\gamma}(z)$ и z находятся на нормали к гиперплоскости $H_c(z)$. Поэтому точка $z \in H_c(z)$ является ортогональной проекцией точки $\hat{\gamma}(z)$ на гиперплоскость $H_c(z)$, и с учетом (3.29) может быть вычислена по известной формуле

$$z = \hat{\gamma}(z) - \frac{\langle c, \hat{\gamma}(z) - z \rangle}{\|c\|^2} c. \quad (3.68)$$

Перепишем это в виде

$$\hat{\gamma}(z) = z + \frac{\langle c, \hat{\gamma}(z) - z \rangle}{\|c\|^2} c. \quad (3.69)$$

Сопоставляя это с (3.48), получаем

$$\beta(z) = \frac{\langle c, \hat{\gamma}(z) - z \rangle}{\|c\|^2}. \quad (3.70)$$

Утверждение доказано.

3.2 Метод поверхностного движения

Метод поверхностного движения строит на поверхности допустимого многогранника путь из произвольной граничной точки $u^{(0)} \in \hat{M} \cap \Gamma(M)$ до точки \bar{x} , являющейся решением задачи ЛП (3.1). Перемещение по поверхности рецессивного многогранника происходит в направлении наибольшего увеличения целевой функции. Реализация метода поверхностного движения приведена в виде алгоритма 3.1. Прокомментируем шаги этого алгоритма. На шаге 1 вводится начальное приближение $u^{(0)}$. Это может быть произвольная граничная точка рецессивного многогранника \hat{M} , удовлетворяющая условию

$$u^{(0)} \in \hat{M} \cap \Gamma(M). \quad (3.71)$$

Для получения хорошего начального приближения может применяться алгоритм, реализующий стадию Quest апекс-метода [85]. На шаге 2 счетчик итераций k устанавливается в значение 0. На шаге 3 строится n -мерный диск D , являющийся пересечением целевой гиперплоскости $H_c(u^{(0)})$, проходящей через точку $u^{(0)}$, и n -мерного шара $\mathcal{B}_r(u^{(0)})$ малого радиуса r с центром в точке $u^{(0)}$. На шаге 4 вычисляется точка $v \in D$ с максимальным смещением относительно границы рецессивного многогранника \hat{M} . Смещение $\beta(z)$ вычисляется с помощью формулы (3.67). Целевая проекция $\hat{\gamma}(z)$, используемая в формуле (3.67), вычисляется с помощью формул (3.49) и (3.50). Смещение $\beta_i(z)$, используемое в формуле (3.49), вычисляется с помощью формулы (3.24). Шаг 5 вычисляет

Листинг 3.1 Метод поверхностного движения

Require: $\hat{H}_i = \{x \in \mathbb{R}^n \mid \langle a_i, x \rangle \leq b_i\}$, $\hat{M} = \bigcap_{i \in \mathcal{I}} \hat{H}_i$, $H_c(z) = \{x \in \mathbb{R}^n \mid \langle c, x - z \rangle = 0\}$

- 1: **input** $u^{(0)}$; **assert** $u^{(0)} \in \hat{M} \cap \Gamma(M)$
- 2: $k := 0$
- 3: $D := H_c(u^{(0)}) \cap \mathcal{B}_r(u^{(0)})$
- 4: $v := \arg \max \{\beta(x) \mid x \in D\}$
- 5: $w := \hat{\gamma}(v)$
- 6: **while** $\langle c, w - u^{(k)} \rangle > \varepsilon_f$ **do**
- 7: **assert** $\exists i \in \mathcal{I} : w, u^{(k)} \in H_i$ ▷ Если не выполняется,
 уменьшить r
- 8: $d := w - u^{(k)}$
- 9: $L = \{u^{(k)} + \lambda d \mid \lambda \in \mathbb{R}_{>0}\}$
- 10: $u^{(k+1)} := \arg \max \{\|x - u^{(k)}\| \mid x \in L \cap \Gamma(M)\}$
- 11: $k := k + 1$
- 12: $D := H_c(u^{(k)}) \cap \mathcal{B}_r(u^{(k)})$
- 13: $v := \arg \max \{\beta(x) \mid x \in D\}$
- 14: $w := \hat{\gamma}(v)$
- 15: **output** $u^{(k)}$
- 16: **stop**

точку w , являющуюся целевой проекцией точки v на границу рецессивного многогранника. Шаги 6–15 реализуют основной аппроксимирующий цикл метода поверхностного движения, геометрическая интерпретация которого приведена на рис. 3.2. Этот цикл выполняется пока справедливо условие

$$\langle c, w - u^{(k)} \rangle > \varepsilon_f, \quad (3.72)$$

где ε_f — малый положительный параметр. Шаг 7 проверяет, что существует рецессивное полупространство \hat{H}_i такое, что граничные точки w и $u^{(k)}$ лежат на гиперплоскости H_i ограничивающей данное полупространство. Это необходимо для того, чтобы перемещение происходило по

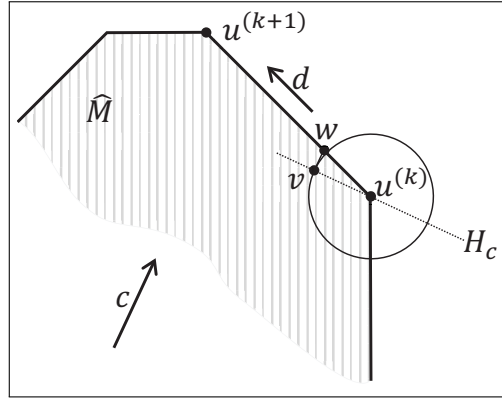


Рисунок 3.2 — Итерация основного цикла в методе поверхностного движения.

поверхности рецессивного многогранника, а не через его внутреннюю область. Если указанное требование не выполняется, необходимо уменьшить радиус r n -мерного шара $\mathcal{B}_r(u^{(k)})$. Подходящий r найдется в силу утверждения 3.10. На шаге 8 формируется вектор d , определяющий направление движения:

$$d = w - u^{(k)}. \quad (3.73)$$

Шаг 9 строит луч L с началом в точке $u^{(k)}$, сонаправленный с вектором d . На шаге 10 определяется следующее приближение $u^{(k+1)}$ как точку на луче L , лежащую на границе допустимого многогранника M и максимально удаленную от точки $u^{(k)}$. По построению из (3.72) следует

$$\langle c, u^{(k)} \rangle < \langle c, u^{(k+1)} \rangle. \quad (3.74)$$

Шаг 11 увеличивает счетчик итераций k на единицу. Шаг 12 строит новый гипердиск D радиуса r с центром в $u^{(k)}$:

$$D := H_c(u^{(k)}) \cap \mathcal{B}_r(u^{(k)}). \quad (3.75)$$

Шаг 13 находит на гипердиске точку v с максимальным смещением. На шаге 14 вычисляется точка w , являющаяся целевой проекцией точки v на границу рецессивного многогранника. На шаге 15 происходит переход на начало основного цикла **while**. Шаг 16 выводит в качестве результата последнее приближение $u^{(k)}$. Шаг 17 завершает работу алгоритма. Отметим,

что по построению алгоритма 3.1 для любого k имеет место

$$u^{(k)} \in M \cap \Gamma(\hat{M}), \quad (3.76)$$

то есть все точки последовательности $\{u^{(k)}\}$, генерируемой алгоритмом 3.1, одновременно лежат и на границе допустимого многогранника M , и на границе рецессивного многогранника \hat{M} .

Следующая теорема гарантирует сходимость алгоритма 3.1 к точному решению задачи ЛП (3.1).

Теорема 3.14. Пусть допустимый многогранник M задачи ЛП (3.1) является ограниченным непустым множеством. Тогда последовательность $\{u^{(k)}\}$, генерируемая алгоритмом 3.1, сходится за конечное число итераций $K \geq 0$ к точному решению \bar{x} задачи ЛП (3.1) при $\varepsilon_f \rightarrow 0$.

Доказательство. Сначала покажем, что количество итераций K , выполняемых в основном цикле алгоритма 3.1, конечно. Предположим противное, то есть алгоритм 3.1 генерирует бесконечную последовательность точек $u^{(k)}$. Но тогда, в силу (3.74), мы получаем бесконечную монотонно возрастающую числовую последовательность

$$\langle c, u^{(0)} \rangle < \langle c, u^{(1)} \rangle < \langle c, u^{(2)} \rangle < \dots \quad (3.77)$$

Так как по условию теоремы допустимый многогранник M является непустым ограниченным множеством, решение \bar{x} задачи ЛП (3.1) существует. В силу (3.76) для всех $k = 0, 1, 2, \dots$ имеем

$$\langle c, u^{(k)} \rangle \leq \langle c, \bar{x} \rangle. \quad (3.78)$$

Это означает, что последовательность (3.77) является ограниченной сверху. По теореме Вейерштрасса монотонно возрастающая ограниченная сверху числовая последовательность имеет конечный предел, равный ее супремуму. То есть существует $K' \in \mathbb{N}$ такой, что

$$\forall k > K' : \langle c, u^{(k+1)} \rangle - \langle c, u^{(k)} \rangle < \varepsilon_f. \quad (3.79)$$

Отсюда следует

$$\forall k > K' : \langle c, w \rangle - \langle c, u^{(k)} \rangle < \varepsilon_f, \quad (3.80)$$

что равносильно

$$\forall k > K' : \langle c, w - u^{(k)} \rangle < \varepsilon_f. \quad (3.81)$$

Получили противоречие с условием (3.72) выполнения цикла, используем на шаге 6 алгоритма 3.1. Таким образом, количество итераций K , выполняемых в основном цикле алгоритма 3.1, конечно.

Пусть точка $u' = u^{(K)}$ является конечной точкой последовательности, генерируемой алгоритмом 3.1. В силу (3.76) имеем

$$u' \in \Gamma(\hat{M}). \quad (3.82)$$

Покажем, что u' является решением задачи ЛП (3.1) при $\varepsilon_f \rightarrow 0$. Рис. 3.3 иллюстрирует последующую часть доказательства. Предположим против-

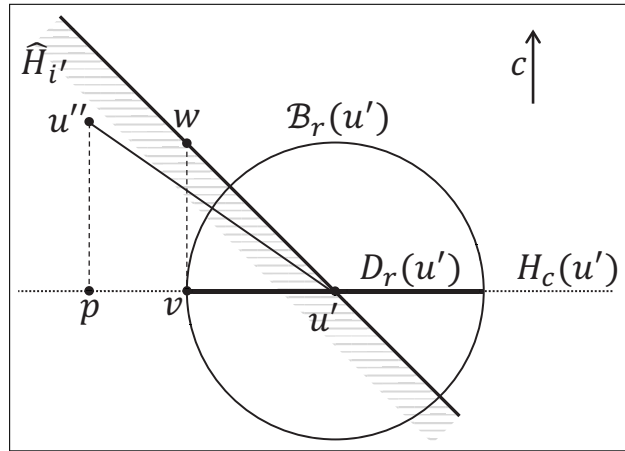


Рисунок 3.3 — Иллюстрация к доказательству теоремы 3.14.

ное, то есть существует точка u'' такая, что

$$u'' \in M, \quad (3.83)$$

и

$$\langle c, u'' \rangle > \langle c, u' \rangle. \quad (3.84)$$

Последнее равносильно

$$\langle c, u'' - u' \rangle > 0. \quad (3.85)$$

Исходя из определения 3.5, вычислим ортогональную проекцию p точки u'' на целевую гиперплоскость $H_c(u')$, проходящую через точку u' :

$$p = u'' - \frac{\langle c, u'' - u' \rangle}{\|c\|^2} c. \quad (3.86)$$

Заметим, что

$$\|p - u'\| \neq 0, \quad (3.87)$$

так как в противном случае, в соответствии с определением 3.7 и формулой (3.82), точка u'' не может принадлежать рецессивному многограннику \hat{M} , что в силу (3.43) противоречит предположению (3.83).

Выберем $r \in \mathbb{R}$, удовлетворяющий условию

$$r > 0, \quad (3.88)$$

для которого существует $i' \in \mathcal{I}$ такой, что

$$\hat{\gamma}(v), u' \in H_{i'} \cap \Gamma(M), \quad (3.89)$$

где

$$v = u' + \frac{r}{\|p - u'\|} (p - u'). \quad (3.90)$$

Это возможно в силу утверждения 3.1. Тогда в соответствии с утверждением 3.12 имеем

$$\hat{\gamma}(v) = \gamma_{i'}(v). \quad (3.91)$$

Обозначим

$$w = \gamma_{i'}(v). \quad (3.92)$$

Согласно утверждению 3.3

$$w = v - \frac{\langle a_{i'}, v \rangle - b_{i'}}{\langle a_{i'}, c \rangle} c. \quad (3.93)$$

Поскольку $u' \in H_{i'}$, из (3.5) следует

$$\langle a_{i'}, u' \rangle = b_{i'}. \quad (3.94)$$

Поэтому (3.93) можно переписать в виде

$$w = v - \frac{\langle a_{i'}, v \rangle - \langle a_{i'}, u' \rangle}{\langle a_{i'}, c \rangle} c. \quad (3.95)$$

Подставив вместо v правую часть формулы (3.90), отсюда получаем

$$w = u' + \frac{r}{\|p - u'\|} (p - u') - \frac{\left\langle a_{i'}, u' + \frac{r}{\|p - u'\|} (p - u') \right\rangle - \langle a_{i'}, u' \rangle}{\langle a_{i'}, c \rangle} c, \quad (3.96)$$

что равносильно

$$w = u' + \frac{r}{\|p - u'\|} (p - u') - \frac{\left\langle a_{i'}, \frac{r}{\|p - u'\|} (p - u') \right\rangle}{\langle a_{i'}, c \rangle} c. \quad (3.97)$$

В соответствии с (3.4) имеем

$$\hat{H}_{i'} = \{x \in \mathbb{R}^n \mid \langle a_{i'}, x \rangle \leq b_{i'}\}. \quad (3.98)$$

Используя (3.94), формулу (3.98) можно переписать в виде

$$\hat{H}_{i'} = \{x \in \mathbb{R}^n \mid \langle a_{i'}, x \rangle \leq \langle a_{i'}, u' \rangle\}. \quad (3.99)$$

Из (3.83) следует $u'' \in \hat{H}_{i'}$. Сопоставляя это с (3.99) получаем

$$\langle a_{i'}, u'' \rangle \leq \langle a_{i'}, u' \rangle, \quad (3.100)$$

что равносильно

$$\langle a_{i'}, u' - u'' \rangle \geq 0. \quad (3.101)$$

Поскольку полупространство $\hat{H}_{i'}$ является рецессивным, в соответствии с утверждением 1 в [85] справедливо

$$\langle a_{i'}, c \rangle > 0. \quad (3.102)$$

В силу (3.97) и (3.86) имеем

$$\begin{aligned}
\langle c, w - u' \rangle &= \left\langle c, \frac{r}{\|p-u'\|} (p - u') - \frac{\left\langle a_{i'}, \frac{r}{\|p-u'\|} (p-u') \right\rangle}{\langle a_{i'}, c \rangle} c \right\rangle \\
&= \frac{r}{\|p-u'\|} \left(\langle c, p - u' \rangle - \frac{\langle a_{i'}, p-u' \rangle}{\langle a_{i'}, c \rangle} \|c\| \right) \\
&= \frac{r}{\|p-u'\|} \left(\left\langle c, u'' - \frac{\langle c, u''-u' \rangle}{\|c\|^2} c - u' \right\rangle - \frac{\left\langle a_{i'}, u'' - \frac{\langle c, u''-u' \rangle}{\|c\|^2} c - u' \right\rangle}{\langle a_{i'}, c \rangle} \|c\| \right) \\
&= \frac{r}{\|p-u'\|} \left(- \frac{\left\langle a_{i'}, u'' - \frac{\langle c, u''-u' \rangle}{\|c\|^2} c - u' \right\rangle}{\langle a_{i'}, c \rangle} \|c\| \right) \\
&= \frac{r\|c\|}{\|p-u'\|\langle a_{i'}, c \rangle} \left(- \left\langle a_{i'}, u'' - \frac{\langle c, u''-u' \rangle}{\|c\|^2} c - u' \right\rangle \right) \\
&= \frac{r\|c\|}{\|p-u'\|\langle a_{i'}, c \rangle} \left(\langle a_{i'}, u' - u'' \rangle + \frac{\langle c, u''-u' \rangle \langle a_{i'}, c \rangle}{\|c\|^2} \right)
\end{aligned}$$

В соответствии с (3.88), (3.87), (3.102), (3.101) и (3.85) отсюда следует

$$\langle c, w - u' \rangle > 0. \quad (3.103)$$

Это означает, что существует $\varepsilon_f > 0$ такой, что

$$\langle c, w - u' \rangle > \varepsilon_f. \quad (3.104)$$

Вспомнив, что $u' = u^{(K)}$, перепишем последнее неравенство в виде

$$\left\langle c, w - u^{(K)} \right\rangle > \varepsilon_f. \quad (3.105)$$

Но тогда, согласно циклу **while**, представленному шагами 6–15 алгоритма 3.1, точка $u^{(K)}$ не может быть конечной точкой последовательности $\{u^{(k)}\}$, генерируемой алгоритмом 3.1. Получили противоречие. Таким образом точка $u' = u^{(K)}$ является решением задачи ЛП (3.1) при $\varepsilon_f \rightarrow 0$.

Теорема доказана.

3.3 Крестообразное рецептивное поле

Определение 3.15. Крестообразным рецептивным полем $\mathfrak{G}_{\text{cross}}(z, \eta, \delta) \subset H_c$ плотности $\delta \in \mathbb{R}_{>0}$ с центром в точке $z \in H_c$ и рангом $\eta \in \mathbb{N}$ будем называть конечное упорядоченное множество точек, расположенных

на осях ортонормированного базиса E_c с центром координат в точке z , определяющего линейное многообразие H_c , с фиксированным расстоянием δ между соседними точками. На каждой полуоси базиса располагается η точек. Центр координат также включается в крестообразное рецептивное поле.

Крестообразное рецептивное поле может быть построено с помощью алгоритма 3.2. Прокомментируем его шаги. На шаге 1 создается пустое множество точек \mathfrak{G} . На шагах 2–19 цикл **for** на каждой итерации заполняет точками одну ось базиса E_c . На шагах 3–18 цикл **for** создает одну точку. На шаге 4 инициализируется точка s , все координаты которой заполнены нулями. Затем цикл **for** на шагах 5–16 вычисляет по очереди каждую координату s_j в базисе E_c и сохраняет ее в s . Если номер координаты не совпадает с номером обрабатываемой оси, то такая координата равна 0 (шаги 6–7). Иначе s_j присваивается координата по оси $e^{(j)}$ (шаги 8–14). Если точка расположена на отрицательной полуоси, то координате по очереди присваиваются значения $\{-\eta, \dots, -1\}$ (шаги 9–10); на положительной полуоси координате присваиваются значения $\{1, \dots, \eta\}$ (шаги 11–13). На шаге 15 вычисленная координата добавляется к s . По окончании записи всех координат в s , на шаге 17 новая точка со сдвигом z добавляется в множество точек рецептивного поля \mathfrak{G} . После того, как множество точек сформировано, на шаге 20 к нему добавляется центральная точка поля. Шаг 21 завершает работу алгоритма.

Пример крестообразного рецептивного поля в пространстве \mathbb{R}^3 приведен на рис. 3.4. Общее количество точек крестообразного поля вычисляется по формуле

$$K_{\text{cross}} = 2\eta(n - 1) + 1. \quad (3.106)$$

Действительно, алгоритм 3.2 в цикле **for** на шагах 2–19 строит точки на каждой из $n - 1$ осей, задаваемых векторами из E_c . На каждой оси во

Листинг 3.2 Построение крестообразного рецептивного поля $\mathfrak{G}_{\text{cross}}(z, \eta, \delta)$

Require: $z \in H_c, \eta \in \mathbb{N}, \delta \in \mathbb{R}_{>0}$

```
1:  $\mathfrak{G} := \emptyset$ 
2: for  $t = 1 \dots n - 1$  do
3:   for  $i = 1 \dots 2\eta$  do
4:      $s := 0$ 
5:     for  $j = 1 \dots n - 1$  do
6:       if  $j \neq t$  then
7:          $s_j = 0$ 
8:       else
9:         if  $i \leq \eta$  then
10:           $s_j := (i - \eta - 1)\delta$ 
11:        else
12:           $s_j := (i - \eta)\delta$ 
13:         $s := s + s_j e^{(j)}$ 
14:       $\mathfrak{G} := \mathfrak{G} \cup \{s + z\}$ 
15:  $\mathfrak{G} := \mathfrak{G} \cup \{z\}$ 
16: stop
```

вложенном цикле **for** (шаги 3–18) от нулевой точки строится η точек в положительном и η точек в отрицательном направлениях. Всего получается $2\eta(n - 1)$ точек. В завершение к получившемуся множеству точек на шаге 20 добавляется центральная точка. В сумме получаем $2\eta(n - 1) + 1$.

Поскольку хранить постоянно весь массив точек нецелесообразно, на практике координаты точки можно вычислять динамически по ее номеру. Именно этот подход используется при построении образа рецептивного поля в алгоритме 3.3.

Дадим краткие комментарии по шагам этого алгоритма. Шаг 1 инициализирует пустое множество \mathfrak{J} , которое будет содержать целевые смещения точек рецептивного поля. Затем цикл **for** перебирает (шаги 2–19)

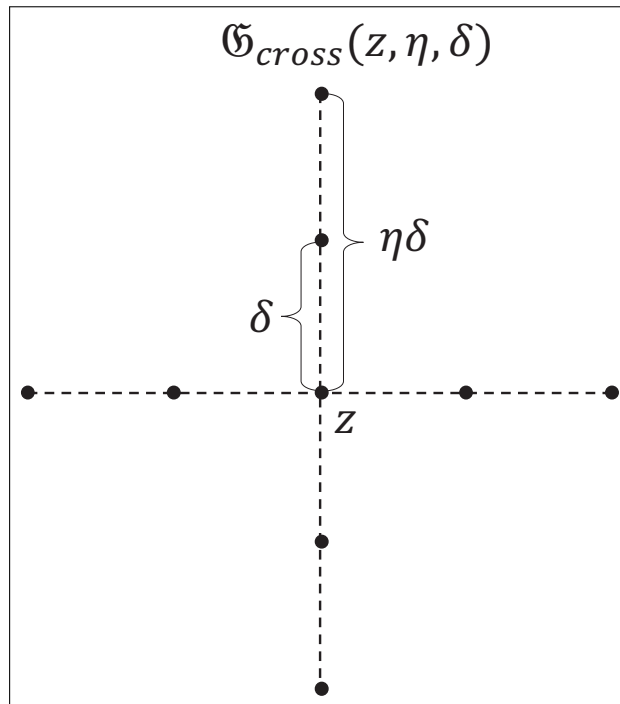


Рисунок 3.4 — Крестообразное поле в пространстве \mathbb{R}^3 .

все точки рецептивного поля, кроме центральной. На шаге 3 вычисляется номер оси, на которой расположена точка. На шаге 4 вычисляется номер точки на оси. На шагах 5–10 вычисляются координаты точки. На шаге 5 в качестве исходных принимаются координаты центральной точки рецептивного поля. В зависимости от того, расположена вычисляемая точка на отрицательной или положительной полуоси (шаг 6), к исходным координатам добавляется перемещение в отрицательную сторону (шаг 7), либо в положительную (шаг 9). Центральная точка на этом этапе не обрабатывается, так как она для всех осей общая. Далее на шагах 11–18 с помощью формулы (3.69) вычисляется целевое смещение полученной точки рецептивного поля относительно границы рецессивного многогранника. После вычисления всех точек рецептивного поля, кроме центральной, на шаге 20 к образу добавляется еще одно смещение, равное нулю, так как центральная точка рецептивного поля всегда располагается на поверхности допустимого многогранника. Следующее утверждение дает оценку временной сложности описанного алгоритма.

Утверждение 3.16. Алгоритм 3.3 имеет временную сложностью $O(n^2m)$.

Листинг 3.3 Построение крестообразного образа $\mathcal{I}_{\text{cross}}(z, \eta, \delta)$

Require: $z \in H_c, \eta \in \mathbb{N}, \delta \in \mathbb{R}_{>0}$

```
1:  $\mathcal{I} := \emptyset$ 
2: for  $k = 1 \dots 2\eta(n - 1)$  do
3:    $l := \lfloor (k - 1) / 2\eta \rfloor + 1$ 
4:    $p := (k - 1) \bmod 2\eta + 1$ 
5:    $g := z$ 
6:   if  $p \leq \eta$  then
7:      $g := g + (p - \eta - 1)\delta e^{(l)}$ 
8:   else
9:      $g := g + (p - \eta)\delta e^{(l)}$ 
10:   $\hat{\beta} := -\frac{\langle a_1, g \rangle - b_1}{\langle a_1, c \rangle} \|c\|$ 
11:  for  $i = 2 \dots m$  do
12:     $\beta_i := -\frac{\langle a_i, g \rangle - b_i}{\langle a_i, c \rangle} \|c\|$ 
13:    if  $\beta_i \leq \hat{\beta}$  then
14:       $\hat{\beta} := \beta_i$ 
15:   $\mathcal{I} := \mathcal{I} \cup \{\hat{\beta}\}$ 
16:  $\mathcal{I} := \mathcal{I} \cup \{0\}$ 
17: stop
```

Доказательство. Рассмотрим алгоритм 3.3. Шаги 5, 7, 9, 11 и 13 имеют временную сложность $O(n)$. Остальные шаги, за исключением операторов цикла **for**, имеют вычислительную сложность $O(1)$. Количество итераций внутреннего цикла **for** (шаг 12) можно оценить как $O(m)$. Следовательно, шаги 12–17 имеют суммарную временную сложность $O(nm)$. Количество итераций внешнего цикла **for** (шаг 2) может быть оценено как $O(n)$. Таким образом, временная сложность алгоритма 3.3 в целом может быть оценена как $O(n^2m)$.

3.4 Построение нейронной сети для движения по гиперплоскости

Пусть в \mathbb{R}^n заданы случайный вектор a и гиперплоскость H_a , ортогональная вектору a . Без ограничения общности мы можем полагать, что гиперплоскость H_a проходит через нулевой вектор:

$$H_a = \{x \in \mathbb{R}^n \mid \langle a, x \rangle = 0\}. \quad (3.107)$$

Зададим произвольный вектор c такой, что

$$\langle a, c \rangle > 0. \quad (3.108)$$

Гиперплоскость H_a представляет собой случайную грань некоторого допустимого многогранника M , а вектор c — градиент случайной целевой функции. При этом полупространство \hat{H}_a , задаваемое формулой

$$\hat{H}_a = \{x \in \mathbb{R}^n \mid \langle a, x \rangle \leq 0\}, \quad (3.109)$$

является рецессивным по отношению к вектору c . Очевидно, что в контексте задачи ЛП $M \subset \hat{H}_a$ и $H_a \cap \Gamma(M) \neq \emptyset$.

Для построения цифрового образа гиперплоскости H_a , используем целевую гиперплоскость в точке 0:

$$H_c(0) = \{x \in \mathbb{R}^n \mid \langle c, x \rangle = 0\}. \quad (3.110)$$

Воспользовавшись (2.43), сформируем базис $E_c \subset H_c(0)$. С помощью алгоритма 4 из [51] построим гиперкубический образ $\mathcal{J}_{\text{cube}}(0, \eta, \delta)$, и с помощью алгоритма 3.3, описанного выше, построим крестообразный образ $\mathcal{J}_{\text{cross}}(0, \eta, \delta)$. Далее мы будем обозначать эти образы как $\mathcal{J}_{\text{cube}}$ и $\mathcal{J}_{\text{cross}}$. С помощью функции

$$\varphi_{\mathcal{J}}(\rho) = 511(\rho - \min(\mathcal{J})) / (\max(\mathcal{J}) - \min(\mathcal{J})) - 256 \quad (3.111)$$

выполним нормализацию обоих образов:

$$\bar{\mathcal{J}}_{\text{cube}} = \{\varphi_{\mathcal{J}_{\text{cube}}}(\rho) \mid \rho \in \mathcal{J}_{\text{cube}}\}, \quad (3.112)$$

$$\bar{\mathcal{J}}_{\text{cross}} = \{ \varphi_{\mathcal{J}_{\text{cross}}}(\rho) \mid \rho \in \mathcal{J}_{\text{cross}} \}. \quad (3.113)$$

Для построения обучающего множества необходимо сопоставить нормализованному образу правильный вектор движения d_a по гиперплоскости H_a . Обозначим через $\pi_a(c)$ ортогональную проекцию вектора c на гиперплоскость H_a :

$$\pi_a(c) = c - \frac{\langle a, c \rangle}{\|a\|^2} a. \quad (3.114)$$

Очевидно, что ортогональная проекция $\pi_a(c)$ указывает направление максимального увеличения значения целевой функции на гиперплоскости H_a . Таким образом

$$d_a = \pi_a(c). \quad (3.115)$$

В качестве правильного ответа вместо n координат вектора d_a в \mathbb{R}^n будем указывать $n - 1$ координату в базисе E_c . Обозначим через $g_a \in H_c$ ортогональную проекцию вектора d_a на гиперплоскость H_c :

$$g_a = d_a - \frac{\langle c, d_a \rangle}{\|c\|^2} c. \quad (3.116)$$

Используя (3.115) и (3.114), данную формулу можно преобразовать к виду

$$g_a = \frac{\langle c, a \rangle}{\|c\|^2} c - a. \quad (3.117)$$

Коэффициентом угла наклона вектора g_a к базисному вектору $e^{(i)}$ назовем косинус угла между векторами g_a и $e^{(i)}$:

$$\cos \alpha_i = \frac{\langle e^{(i)}, g_a \rangle}{\|g_a\|}. \quad (3.118)$$

Взятые вместе, коэффициенты углов наклона образуют вектор правильного ответа

$$y_a = \left(\frac{\langle e^{(1)}, g_a \rangle}{\|g_a\|}, \dots, \frac{\langle e^{(n-1)}, g_a \rangle}{\|g_a\|} \right). \quad (3.119)$$

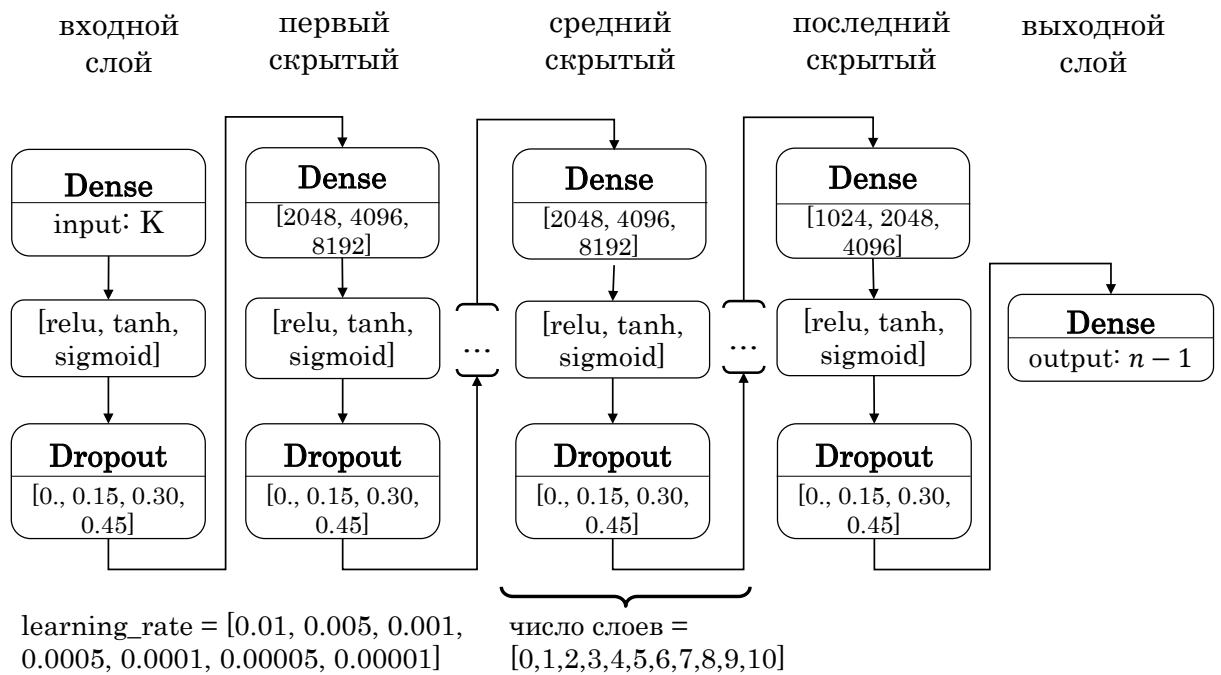


Рисунок 3.5 — Архитектура гипермодели

Для настройки оптимальных гиперпараметров нейронной сети была разработана гипермодель, представленная на рис. 3.5. Эта гипермодель включает входной слой, блок скрытых слоев и выходной слой. Блок скрытых слоев состоит из первого скрытого слоя, переменного числа средних скрытых слоев и последнего скрытого слоя. Все слои являются полностью связанными (dense connection). Выбор функции активации для всех слоев осуществлялся из набора $\{ReLU, sigmoid, tanh\}$. Входной слой имеет K нейронов, соответствующих точкам используемого рецептивного поля. Выходной слой имеет $n - 1$ нейронов, соответствующих числу коэффициентов углов наклона вектора g_a .

Подбор гиперпараметров выполнялся на обучающем множестве, сгенерированном в пространстве \mathbb{R}^{10} с конфигурацией рецептивного поля, представленной в табл. 6.

Таблица 6 — Параметры рецептивного поля

Параметр	Семантика	Значение
n	размерность пространства	10
η	ранг рецептивного поля	5
δ	расстояние между соседними точками	1
K_{cube}	число точек рецептивного поля	91

Для генерации случайных координат векторов a и c использовалось стандартное нормальное распределение. Число нейронов для скрытых слоев подбиралось из набора $\{1024, 2048, 4096, 8192\}$. Количество средних скрытых слоев подбиралось из набора $\{0, 1, \dots, 8\}$. Заметим, что максимально возможное число скрытых слоев при таком выборе параметров равнялось 10, что соответствует заданной размерности пространства $n = 10$.

Основываясь на представленной гипермодели, был выполнен байесовский поиск оптимального набора гиперпараметров с использованием платформы W&B [86]. При обучении был использован оптимизатор *RMSProp* [87]. Размер блока обучающих прецедентов (batch size) равнялся 128. В качестве функции потерь (loss) была использована косинусовая мера (cosine similarity)

$$CS = - \frac{\sum_{k=1}^{n-1} (\alpha_k \cdot y_k)}{\sqrt{\sum_{k=1}^{n-1} \alpha_k^2} \cdot \sqrt{\sum_{k=1}^{n-1} y_k^2}}. \quad (3.120)$$

Здесь α_k — это значения коэффициентов наклона, предсказанные нейронной сетью, y_k — коэффициенты наклона, рассчитанные на основе формулы (3.119). Набор обучающих данных из 100 000 прецедентов был разделен следующим образом:

- обучающая выборка: 80000 элементов;
- тестовая выборка: 15000 элементов;
- валидационная выборка: 5000 элементов.

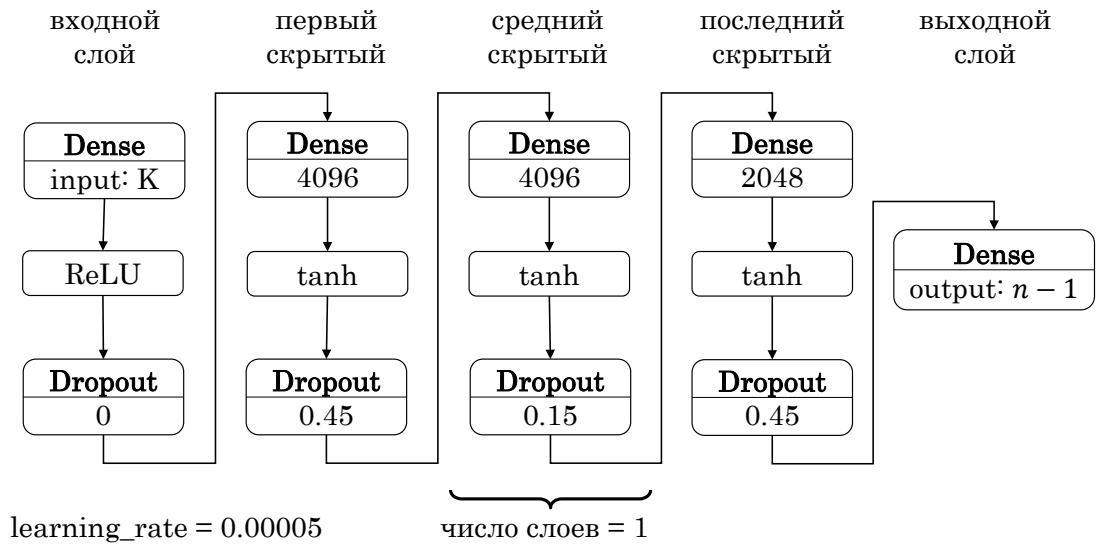


Рисунок 3.7 — Архитектура нейронной сети

Для обучения и тестирования нейронных сетей использовался комплекс «Нейрокомпьютер» Южно-Уральского государственного университета [88], оборудованный графическими процессорами nVidia Tesla V100. Обучение производилось при помощи библиотек `keras` и `TensorFlow`. В результате была получена глубокая ИНС, архитектура которой представлена на рис. 3.7. Для оценки качества работы нейронных сетей была использована оригинальная модификация средней абсолютной ошибки, получившая название «средняя абсолютная нормализованная ошибка» MANE (mean absolute normalized error):

$$\text{MANE} = \frac{1}{n} \cdot \sum_{i=1}^n \left| \frac{y_i}{\|y\|} - \frac{\alpha_i}{\|\alpha\|} \right|. \quad (3.121)$$

Создание обучающего множества и обучение искусственной нейронной сети производилось по схеме, представленной на рис. 3.8. Сначала программа генерации случайных гиперплоскостей, приняв в качестве входного параметра число Q , генерирует Q пар $\{a, c\}$, где каждая пара состоит из случайного вектора a , определяющего гиперплоскость (3.107) и случайного вектора целевой функции c . Массив сгенерированных данных помещается в файл, передаваемый программе визуализации. Визуализатор принимает входные параметры, определяющие положение точек рецептивного поля:

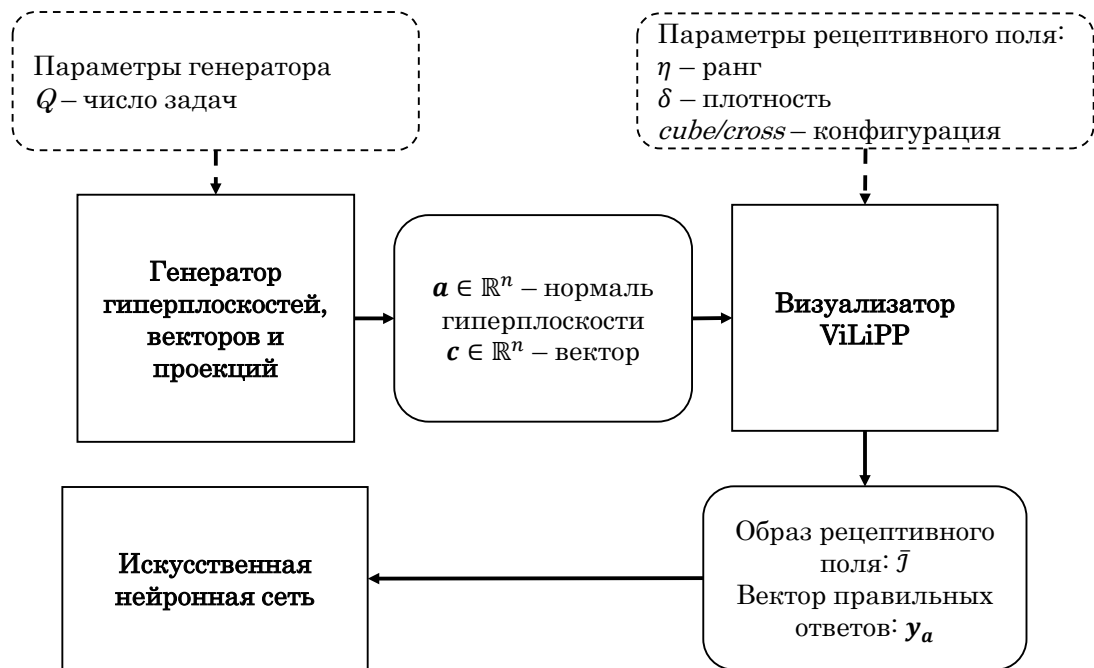


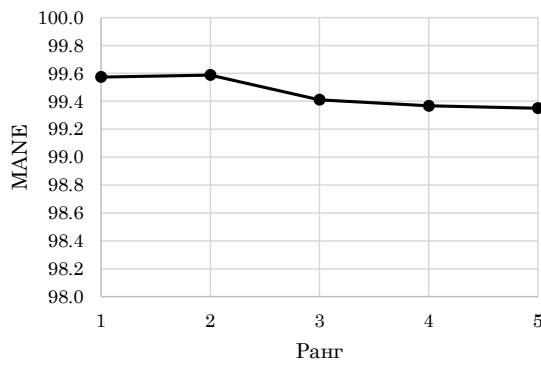
Рисунок 3.8 — Архитектура программного комплекса для построения обучающего множества

- ранг рецептивного поля η ;
- плотность рецептивного поля δ ;
- форма рецептивного поля `cube` (гиперкубическое) или `cross` (крестообразное).

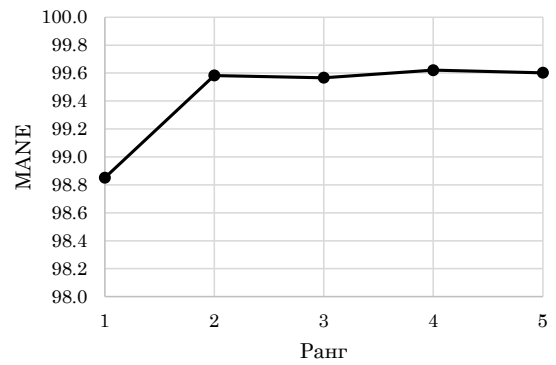
В результате работы визуализатора формируется файл прецедентов, содержащий Q строк. В каждой строке записаны K нормализованных по формуле (3.111) коэффициентов, формирующих образ \bar{J} , и $n - 1$ коэффициентов угла наклона, формирующих вектор правильного ответа y_a .

3.5 Вычислительные эксперименты

В первой серии экспериментов исследовалось влияние конфигурации и числа точек рецептивного поля на точность работы ИНС. На основе архитектуры, представленной на рис. 3.7, были построены 10 нейронных сетей, отличающихся числом входных нейронов. Число входных нейронов соответствовало гиперкубической и крестообразной конфигурациям рецептивного поля. Для каждой конфигурации перебиралось значение ранга η от 1 до 5. Результаты представлены на рис. 3.9. На графиках видно,



а) гиперкубическое

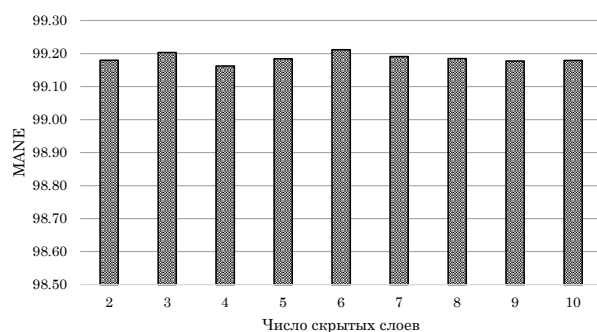


б) крестообразное

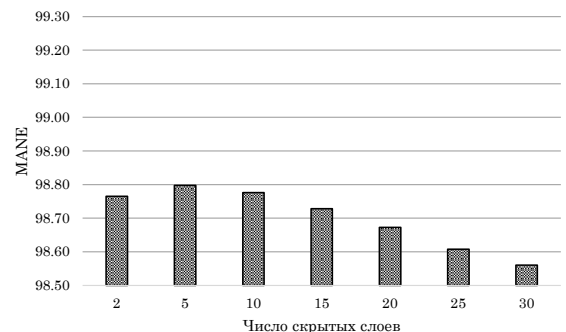
Рисунок 3.9 — Зависимость точности от конфигурации и ранга рецептивного поля в \mathbb{R}^4

что крестообразное рецептивное поле незначительно уступает в точности гиперкубическому. При этом для размерности $n = 4$ наилучшие результаты крестообразное поле демонстрирует, начиная с ранга 2 (рис. 3.9б). Гиперкубическое рецептивное поле для этой размерности демонстрирует наилучший результат также при ранге 2, однако дальнейшее увеличение ранга приводит к ухудшению точности (рис. 3.9а). Основываясь на полученных результатах, для дальнейших экспериментов с задачами больших размерностей было выбрано крестообразное рецептивное поле ранга 5.

Во второй серии вычислительных экспериментов была исследована зависимость точности ИНС от числа скрытых слоев. Для пространств \mathbb{R}^{10} и \mathbb{R}^{30} был построен набор нейронных сетей на основе архитектуры, представленной на рис. 3.7, для которых число скрытых слоев варьировалось от 2 до n . Результаты представлены на рис. 3.10. Для размерности



а) в пространстве \mathbb{R}^{10}



б) в пространстве \mathbb{R}^{30}

Рисунок 3.10 — Зависимость точность ИНС от числа скрытых слоев

10 число скрытых слоев практически не влияет на точность работы ИНС (рис. 3.10а). Однако для размерности 30 лучший результат показывает ИНС с пятью скрытыми слоями (рис. 3.10б).

В третьей серии экспериментов была предпринята попытка улучшить точность работы ИНС за счет увеличения времени обучения. Были протестированы ИНС для пространств \mathbb{R}^{10} и \mathbb{R}^{30} . Обучение нейронной сети в \mathbb{R}^{10} происходило на протяжении 900 эпох, в пространстве \mathbb{R}^{30} — 800 эпох. И в том и в другом случае время обучения составило 60 мин. Результаты представлены на рис. 3.11. Обращает на себя внимание тот факт, что в \mathbb{R}^{10} максимальная точность достигается на 400 эпохе и далее существенно уже не меняется. В \mathbb{R}^{30} точность увеличивалась практически линейно вплоть до 800 эпохи. Таким образом, можно сделать предположение, что с ростом размерности пространства время обучения ИНС существенно увеличивается.

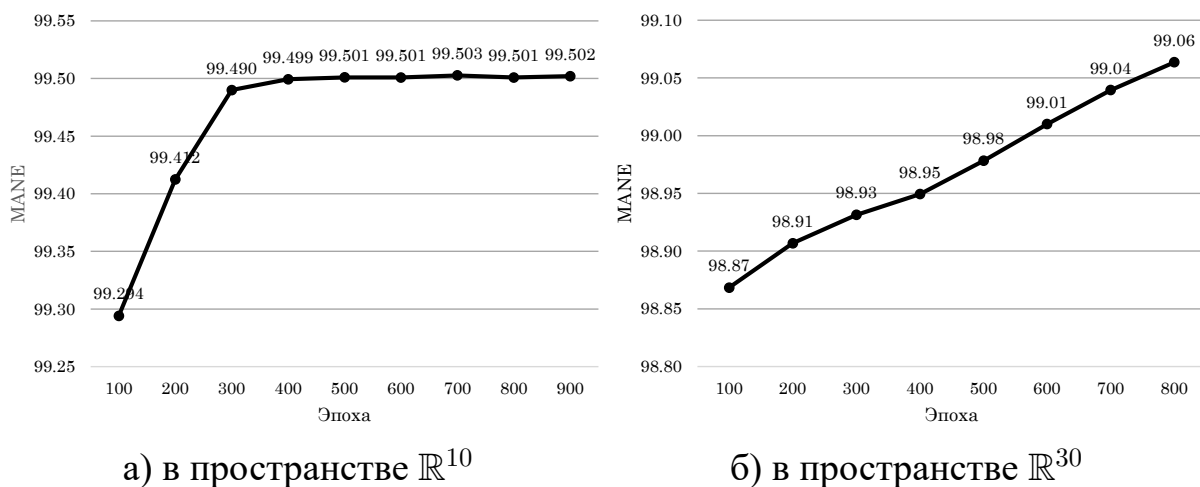


Рисунок 3.11 — Зависимость точность ИНС от количество эпох обучения

ЗАКЛЮЧЕНИЕ

В ходе работы были достигнуты результаты, перечисленные ниже.

- Исследованы имеющиеся итерационные методы решения задачи ЛП. На их основе разработан новый метод решения многомерной задачи ЛП, подходящий для обучения искусственных нейронных сетей.
- Разработан метод визуализации многомерных задач ЛП. Реализован алгоритм параллельных вычислений на его основе, обладающий высокой масштабируемостью.
- Построено обучающее множество для искусственной нейронной сети. Проведены вычислительные эксперименты с обученной ИНС.
- Разработан новый итерационный метод решения многомерных задач ЛП на основе комбинации визуального представления задачи ЛП и нейронной сети прямого распространения.

В настоящий момент ведется разработка экспериментального программного комплекса, позволяющего решать многомерные задачи линейного программирования. В дальнейшем разработанный комплекс может быть интегрирован в прикладные коммерческие пакеты, предназначенные для решения задач управления и оптимизации в промышленности и бизнесе.

В рамках работы были опубликованы следующие статьи:

1. Olkhovsky N. A., Sokolinsky L. B. Visualizing Multidimensional Linear Programming Problems. — 2022. — URL: https://link.springer.com/10.1007/978-3-031-11623-0_13
2. Ольховский Н.А., Соколинский Л.Б. Визуальное представление многомерных задач линейного программирования // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2022. Т. 11, № 1. С. 31–56. DOI: 10.14529/cmse220103

Получено свидетельство о регистрации программы для ЭВМ:

- ViLiPP: Визуализатор многомерных задач линейного программирования [Текст] : Свидетельство о государственной регистрации программы для ЭВМ RU2022664550 Рос. Федерация / Ольховский Н. А., Соколинский Л. Б. (RU) ; правообладатель ФГАОУ ВО «ЮУрГУ (НИУ)» (RU) ; Федеральная служба по интеллектуальной собственности. — № 2022663846 ; опубл. 01.08.2022, Бюл. № 8. Ольховский Н.А., Соколинский Л.Б.

Представлены доклады на международных конференциях:

1. Global Smart Industry Conference (GloSIC'2020), South Ural State University, November 17–19, 2020, Chelyabinsk.
2. Parallel Computational Technologies (PCT'2022), Joint Institute for Nuclear Research, March 29-31, 2022, Dubna.
3. Parallel Computational Technologies (PCT'2023), ITMO University, March 28-30, 2023, Saint Petersburg.

В заключение автор выражает благодарность и большую признательность научному руководителю Соколинскому Л. Б. за поддержку, помощь, обсуждение результатов и научное руководство. Автор благодарит Силкину Н. С. и весь коллектив кафедры системного программирования ЮУрГУ за возможность вести научную работу и готовить диссертацию в спокойной и продуктивной атмосфере. Автор также благодарит свою жену Ситницких И. О. и всех, кто сделал настоящую работу автора возможной.

ЛИТЕРАТУРА

1. *Jagadish H. V., Gehrke J., Labrinidis A., Papakonstantinou Y., Patel J. M., Ramakrishnan R., Shahabi C.* Big data and its technical challenges // *Communications of the ACM*. — 2014. — Июль. — Т. 57, вып. 7. — С. 86—94. — URL: <https://dl.acm.org/doi/10.1145/2611567>.
2. *Hartung T.* Making Big Sense From Big Data // *Frontiers in Big Data*. — 2018. — Т. 1. — С. 5.
3. *Sokolinskaya I., Sokolinsky L. B.* On the Solution of Linear Programming Problems in the Age of Big Data // / под ред. L. Sokolinsky, M. Zymbler. — Springer, 2017. — С. 86—100.
4. *Chung W.* Applying large-scale linear programming in business analytics // 2016—January. — IEEE, 12.2015. — С. 1860—1864. — URL: <http://ieeexplore.ieee.org/document/7385970/>.
5. *Gondzio J., Gruca J. A., Hall J. J., Laskowski W., Żukowski M.* Solving large-scale optimization problems related to Bell's Theorem // *Journal of Computational and Applied Mathematics*. — 2014. — Июнь. — Т. 263. — С. 392—404. — URL: <https://linkinghub.elsevier.com/retrieve/pii/S0377042713006730>.
6. *Sodhi M. S.* LP Modeling for Asset-Liability Management: A Survey of Choices and Simplifications // *Operations Research*. — 2005. — Апр. — Т. 53, вып. 2. — С. 181—196. — URL: <https://pubsonline.informs.org/doi/10.1287/opre.1040.0185>.
7. *Branke J.* Optimization in Dynamic Environments. — 2002. — URL: http://link.springer.com/10.1007/978-1-4615-0911-0_2.
8. *Brogaard J., Hendershott T., Riordan R.* High-Frequency Trading and Price Discovery // *Review of Financial Studies*. — 2014. — Август. — Т. 27, вып. 8. — С. 2267—2306. — URL: <https://academic.oup.com/rfs/article-lookup/doi/10.1093/rfs/hhu032>.

9. *Deng S., Huang X., Wang J., Qin Z., Fu Z., Wang A., Yang T.* A Decision Support System for Trading in Apple Futures Market Using Predictions Fusion // IEEE Access. — 2021. — T. 9. — C. 1271—1285. — URL: <https://ieeexplore.ieee.org/document/9306753/>.
10. *Seregin G.* Lecture Notes on Regularity Theory for the Navier-Stokes Equations. — WORLD SCIENTIFIC, 11.2014. — URL: <https://www.worldscientific.com/worldscibooks/10.1142/9314>.
11. *Demin D.* Synthesis of optimal control of technological processes based on a multialternative parametric description of the final state // Eastern-European Journal of Enterprise Technologies. — 2017. — Т. 3, вып. 4—87.
12. *Kazarinov L. S., Shnayder D. A., Kolesnikova O. V.* Heat load control in steam boilers // — IEEE, 05.2017. — C. 1—4. — URL: <https://ieeexplore.ieee.org/document/8076177/>.
13. *Zagoskina E., Barbasova T., Shnaider D.* Intelligent Control System of Blast-furnace Melting Efficiency // — IEEE, 10.2019. — C. 0710—0713. — URL: <https://ieeexplore.ieee.org/document/8958221/>.
14. *Fleming J., Yan X., Allison C., Stanton N., Lot R.* Real-time predictive eco-driving assistance considering road geometry and long-range radar measurements // IET Intelligent Transport Systems. — 2021. — Т. 15, вып. 4. — C. 573—583.
15. *Scholl M., Minnerup K., Reiter C., Bernhardt B., Weisbrodt E., Newiger S.* Optimization of a thermal management system for battery electric vehicles // — IEEE, 2019.
16. *Meisel S.* Dynamic Vehicle Routing. — 2011.
17. *Cheng A. M. K.* Real-Time Scheduling and Schedulability Analysis. — 2002.
18. *Kopetz H.* Real-Time Scheduling. — 2011.

19. *Dantzig G. B.* Linear programming and extensions. — Princeton university press, 1998. — С. 656.
20. *Hall J. A. J., McKinnon K. I. M.* Hyper-sparsity in the revised simplex method and how to exploit it // Computational Optimization and Applications. — 2005. — Т. 32, ВЫП. 3. — С. 259—283.
21. *Klee V., Minty G. J.* How good is the simplex algorithm? // / под ред. О. Shisha. — Academic Press, 1972. — С. 159—175.
22. *Jeroslow R. G.* The simplex algorithm with the pivot rule of maximizing criterion improvement // Discrete Mathematics. — 1973. — Т. 4, ВЫП. 4. — С. 367—377. — URL: <https://www.sciencedirect.com/science/article/pii/0012365X73901714?via%3Dihub>.
23. *Zadeh N.* A bad network problem for the simplex method and other minimum cost flow algorithms // Mathematical Programming. — 1973. — Т. 5, ВЫП. 1. — С. 255—266.
24. *Bartels R. H., Stoer J., Zenger C.* A Realization of the Simplex Method Based on Triangular Decompositions. — 1971.
25. *Tolla P.* A Survey of Some Linear Programming Methods. — 2014.
26. *Hall J. A. J.* Towards a practical parallelisation of the simplex method // Computational Management Science. — 2010. — Т. 7, ВЫП. 2. — С. 139—170.
27. *Mamalis B., Pantziou G.* Advances in the Parallelization of the Simplex Method. — 2015.
28. *Lubin M., Hall J. A. J., Petra C. G., Anitescu M.* Parallel distributed-memory simplex for large-scale stochastic LP problems // Computational Optimization and Applications. — 2013. — Т. 55, ВЫП. 3. — С. 571—596. — URL: <http://link.springer.com/10.1007/s10589-013-9542-y>.

29. *Khachiyan L. G.* Polynomial algorithms in linear programming // USSR Computational Mathematics and Mathematical Physics. — 1980. — Т. 20, вып. 1. — С. 53—72. — URL: <https://www.sciencedirect.com/science/article/abs/pii/0041555380900610?via%3Dihub>.
30. *Shor N. Z.* Cut-off method with space extension in convex programming problems // Cybernetics and Systems Analysis. — 1977. — Т. 13, вып. 1. — С. 94—96.
31. *Nesterov Y., Nemirovskii A.* Interior-Point Polynomial Algorithms in Convex Programming. — Society for Industrial, Applied Mathematics, 01.1994. — С. ix + 396. — URL: <http://epubs.siam.org/doi/book/10.1137/1.9781611970791>.
32. *Karmarkar N.* A new polynomial-time algorithm for linear programming // Combinatorica. — 1984. — Т. 4, вып. 4. — С. 373—395.
33. *Gondzio J.* Interior point methods 25 years later // European Journal of Operational Research. — 2012. — Т. 218, вып. 3. — С. 587—601.
34. *Fathi-Hafshejani S., Mansouri H., Peyghami M. R., Chen S.* Primal–dual interior-point method for linear optimization based on a kernel function with trigonometric growth term // Optimization. — 2018. — Окт. — Т. 67, вып. 10. — С. 1605—1630. — URL: <https://www.tandfonline.com/doi/full/10.1080/02331934.2018.1482297>.
35. *Asadi S., Mansouri H.* A Mehrotra type predictor-corrector interior-point algorithm for linear programming // Numerical Algebra, Control and Optimization. — 2019. — Т. 9, вып. 2. — С. 147—156.
36. *Yuan Y.* Implementation tricks of interior-point methods for large-scale linear programs //. — International Society for Optics, Photonics, 2011.
37. *Kheirfam B., Haghghi M.* A full-Newton step infeasible interior-point method for linear optimization based on a trigonometric kernel function // Optimization. — 2016. — Апр. — Т. 65, вып. 4. — С. 841—857.

38. *Xu Y., Zhang L., Zhang J.* A full-modified-newton step infeasible interior-point algorithm for linear optimization // *Journal of Industrial and Management Optimization.* — 2016. — Т. 12, вып. 1. — С. 103—116.
39. *Roos C., Terlaky T., Vial J.-P.* Interior Point Methods for Linear Optimization. — Springer, 2005. — С. 500. — URL: <http://link.springer.com/10.1007/b100325>.
40. *Sokolinskaya I.* Parallel Method of Pseudoprojection for Linear Inequalities. — 04.2018.
41. *Eremin I. I.* Methods of fejer's approximations in convex programming // *Mathematical Notes of the Academy of Sciences of the USSR.* — 1968. — Февр. — Т. 3, вып. 2. — С. 139—149. — URL: <http://link.springer.com/10.1007/BF01094336>.
42. *Gondzio J., Grothey A.* Direct Solution of Linear Systems of Size 109 Arising in Optimization with Interior Point Methods // 3911 LNCS / под ред. R. Wyrzykowski, J. Dongarra, N. Meyer, J. Wasniewski. — Springer, 2006. — С. 513—525. — URL: http://link.springer.com/10.1007/11752578_62.
43. *Prieto A., Prieto B., Ortigosa E. M., Ros E., Pelayo F., Ortega J., Rojas I.* Neural networks: An overview of early research, current frameworks and new challenges // *Neurocomputing.* — 2016. — Т. 214. — С. 242—268.
44. *Raina R., Madhavan A., Ng A. Y.* Large-scale deep unsupervised learning using graphics processors // — ACM Press, 2009. — С. 873—880. — URL: <http://portal.acm.org/citation.cfm?doid=1553374.1553486>.
45. *Tank D. W., Hopfield J. J.* Simple 'neural' optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit // *IEEE transactions on circuits and systems.* — 1986. — Т. CAS—33, вып. 5. — С. 533—541.

46. *Kennedy M. P., Chua L. O.* Unifying the Tank and Hopfield Linear Programming Circuit and the Canonical Nonlinear Programming Circuit of Chua and Lin // *IEEE Transactions on Circuits and Systems*. — 1987. — Т. 34, ВЫП. 2. — С. 210—214.
47. *Rodriguez-Vazquez A., Dominguez-Castro R., Rueda A., Huertas J., Sanchez-Sinencio E.* Nonlinear switched capacitor 'neural' networks for optimization problems // *IEEE Transactions on Circuits and Systems*. — 1990. — Март. — Т. 37, ВЫП. 3. — С. 384—398. — URL: [http : //ieeexplore.ieee.org/document/52732/](http://ieeexplore.ieee.org/document/52732/).
48. *Zak S. H., Upatising V.* Solving Linear Programming Problems with Neural Networks: A Comparative Study // *IEEE Transactions on Neural Networks*. — 1995. — Т. 6, ВЫП. 1. — С. 94—104.
49. *Malek A., Yari A.* Primal–dual solution for the linear programming problems using neural networks // *Applied Mathematics and Computation*. — 2005. — Т. 167, ВЫП. 1. — С. 198—211.
50. *Liu X., Zhou M.* A one-layer recurrent neural network for non-smooth convex optimization subject to linear inequality constraints // *Chaos, Solitons and Fractals*. — 2016. — Т. 87. — С. 39—46.
51. *Olkhovsky N. A., Sokolinsky L. B.* Visualizing Multidimensional Linear Programming Problems. — 2022. — URL: https://link.springer.com/10.1007/978-3-031-11623-0_13.
52. *LeCun Y., Bengio Y., Hinton G.* Deep learning // *Nature*. — 2015. — Т. 521, ВЫП. 7553. — С. 436—444.
53. *Lachhwani K.* Application of Neural Network Models for Mathematical Programming Problems: A State of Art Review // *Archives of Computational Methods in Engineering*. — 2020. — Т. 27. — С. 171—182.

54. *Kaczmarz S.* Angenherete Auflsung von Systemen linearer Gleichungen // Bulletin International de l'Acadmie Polonaise des Sciences et des Lettres. Classe des Sciences Mathmatiques et Naturelles. Srie A, Sciences Mathmatiques. — 1937. — T. 35. — C. 355—357.
55. *Kaczmarz S.* Approximate solution of systems of linear equations // International Journal of Control. — 1993. — ИЮНЬ. — Т. 57, ВЫП. 6. — C. 1269—1271. — URL: <https://www.tandfonline.com/doi/abs/10.1080/00207179308934446>.
56. *Cimmino G.* Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari // La Ricerca Scientifica, XVI, Series II, Anno IX, 1. — 1938. — C. 326—333.
57. *Gastinel N.* Linear Numerical Analysis. — Academic Press, 1971. — C. ix+341.
58. *Agmon S.* The relaxation method for linear inequalities // Canadian Journal of Mathematics. — 1954. — Т. 6. — C. 382—392.
59. *Motzkin T. S., Schoenberg I. J.* The relaxation method for linear inequalities // Canadian Journal of Mathematics. — 1954. — Т. 6. — C. 393—404.
60. *Censor Y., Elfving T.* New methods for linear inequalities // Linear Algebra and its Applications. — 1982. — Т. 42. — C. 199—211. — URL: <http://www.sciencedirect.com/science/article/pii/0024379582901495%20http://linkinghub.elsevier.com/retrieve/pii/0024379582901495>.
61. *Pierro A. R. D., Iusem A. N.* A simultaneous projections method for linear inequalities // Linear Algebra and its Applications. — 1985. — Т. 64. — C. 243—253.
62. *Sokolinskaya I., Sokolinsky L.* Revised Pursuit Algorithm for Solving Non-stationary Linear Programming Problems on Modern Computing Clusters

- with Manycore Accelerators. — 2016. — URL: http://link.springer.com/10.1007/978-3-319-55669-7_17.
63. *Sokolinskaya I. M., Sokolinsky L. B.* Scalability Evaluation of Cimmino Algorithm for Solving Linear Inequality Systems on Multiprocessors with Distributed Memory // *Supercomputing Frontiers and Innovations*. — 2018. — Т. 5, вып. 2. — С. 11—22.
64. *Sokolinsky L. B., Sokolinskaya I. M.* Scalable Parallel Algorithm for Solving Non-stationary Systems of Linear Inequalities // *Lobachevskii Journal of Mathematics*. — 2020. — Август. — Т. 41, вып. 8. — С. 1571—1580. — URL: <https://link.springer.com/10.1134/S1995080220080181>.
65. *Gonzalez-Gutierrez E., Todorov M. I.* A relaxation method for solving systems with infinitely many linear inequalities // *Optimization Letters*. — 2012. — Февр. — Т. 6, вып. 2. — С. 291—298. — URL: <http://link.springer.com/10.1007/s11590-010-0244-4>.
66. *Vasin V. V., Eremin I. I.* Operators and Iterative Processes of Fejer Type. Theory and Applications. — Walter de Gruyter, 01.2009. — С. 155. — URL: <https://www.degruyter.com/view/books/9783110218190/9783110218190/9783110218190.xml>.
67. *Eremin I. I., Popov L. D.* Fejér processes in theory and practice: Recent results // *Russian Mathematics*. — 2009. — Янв. — Т. 53, вып. 1. — С. 36—55. — URL: <http://link.springer.com/10.3103/S1066369X09010022>.
68. *Nurminski E. A.* Single-projection procedure for linear optimization // *Journal of Global Optimization*. — 2016. — Т. 66, вып. 1. — С. 95—110. — URL: <https://link.springer.com/article/10.1007/s10898-015-0337-9>.
69. *Censor Y.* Can linear superiorization be useful for linear optimization problems? // *Inverse Problems*. — 2017. — Т. 33, вып. 4. — С. 44006. —

URL: <http://stacks.iop.org/0266-5611/33/i=4/a=044006?key=crossref.6a47b1c246620584482ca93e603bce1a>.

70. *Sokolinsky L. B., Sokolinskaya I. M.* Scalable Method for Linear Optimization of Industrial Processes // — IEEE, 11.2020. — С. 20—26. — URL: <https://ieeexplore.ieee.org/document/9267854/>.
71. *Gould N. I. M.* How good are projection methods for convex feasibility problems? // Computational Optimization and Applications. — 2008. — Май. — Т. 40, вып. 1. — С. 1—12. — URL: <http://link.springer.com/10.1007/s10589-007-9073-5>.
72. *Sokolinsky L. B.* BSF: A parallel computation model for scalability estimation of iterative numerical algorithms on cluster computing systems // Journal of Parallel and Distributed Computing. — 2021. — Т. 149. — С. 193—206.
73. *Sokolinsky L. B.* BSF-skeleton: A Template for Parallelization of Iterative Numerical Algorithms on Cluster Computing Systems // MethodsX. — 2021. — Т. 8. — Article number 101437.
74. *Dolganina N., Ivanova E., Bilenko R., Rekachinsky A.* HPC Resources of South Ural State University // / под ред. L. Sokolinsky, M. Zymbler. — Springer, 2022. — С. 43—55.
75. *Sokolinsky L. B., Sokolinskaya I. M.* FRaGenLP: A Generator of Random Linear Programming Problems for Cluster Computing Systems // / под ред. L. Sokolinsky, M. Zymbler. — Springer, 2021. — С. 164—177.
76. *Sokolinsky L. B., Sokolinskaya I. M.* VaLiPro: Linear Programming Validator for Cluster Computing Systems // Supercomputing Frontiers and Innovations. — 2021. — Т. 8, вып. 3. — С. 51—61.

77. *Ежова Н. А., Соколинский Л. Б.* Модель параллельных вычислений BSF-MR // Системы управления и информационные технологии. — 2019. — 3(77). — С. 15—21.
78. *Bird R. S.* Lectures on Constructive Functional Programming // Constructive Methods in Computing Science. NATO ASI Series F: Computer and Systems Sciences, vol. 55 / под ред. М. Broy. — Berlin, Heidelberg : Springer, 1988. — С. 151—216.
79. *Sokolinsky L. B.* BSF-skeleton. — 2019. — URL: <https://github.com/leonid-sokolinsky/BSF-skeleton> (дата обр. 24.01.2022).
80. *Gropp W.* MPI 3 and Beyond: Why MPI Is Successful and What Challenges It Faces // Recent Advances in the Message Passing Interface. EuroMPI 2012. Lecture Notes in Computer Science, vol. 7490 / под ред. J. Traff, S. Benkner, J. Dongarra. — Berlin, Heidelberg : Springer, 2012. — С. 1—9.
81. *Kale V.* Shared-memory Parallel Programming with OpenMP // Parallel Computing Architectures and APIs. — Boca Raton : Chapman, Hall/CRC, 2019. — Гл. 14. С. 213—222.
82. *Kostenetskiy P., Semenikhina P.* SUSU Supercomputer Resources for Industry and fundamental Science // Proceedings - 2018 Global Smart Industry Conference, GloSIC 2018, art. no. 8570068. — IEEE, 2018. — С. 7.
83. *Соколинский Л. Б., Соколинская И. М.* О генерации случайных задач линейного программирования на кластерных вычислительных системах // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. — 2021. — Т. 10, № 1. — С. 38—52.

84. *Соколинский Л. Б.* Свидетельство о государственной регистрации программы для ЭВМ № 2021619526 Российская Федерация. Генератор случайных задач линейного программирования FRaGenLP : № 2021618165 : заявл. 28.05.2021 : опубл. 10.06.2021.
85. *Sokolinsky L. B., Sokolinskaya I. M.* Apex Method: A New Scalable Iterative Method for Linear Programming // *Mathematics*. — 2023. — Март. — Т. 11, вып. 7. — С. 1654. — URL: <https://www.mdpi.com/2227-7390/11/7/1654>.
86. *Biewald L.* Experiment Tracking with Weights & Biases // Software available from wandb. com. — 2020. — January.
87. *Goodfellow I., Bengio Y., Courville A.* Deep Learning (Adaptive Computation and Machine Learning). Т. 8. — 2016.
88. *Bilenko R. V., Dolganina N. Y., Ivanova E. V., Rekachinsky A. I.* High-performance Computing Resources of South Ural State University // *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering*. — 2022. — Т. 11, № 1. — С. 15—30. — URL: <https://vestnik.susu.ru/cmi/article/view/11878>.