На правах рукописи

Нгуен Минь Ханг

РАЗРАБОТКА И РЕАЛИЗАЦИЯ ЧИСЛЕННЫХ МЕТОДОВ РЕШЕНИЯ ОПТИМИЗАЦИОННЫХ ЗАДАЧ БОЛЬШОЙ РАЗМЕРНОСТИ

01.01.09 — Дискретная математика и математическая кибернетика

Автореферат

диссертации на соискание ученой степени

кандидата физико-математических наук

Москва — 2009

Работа выполнена в Учреждении Российской академии наук Вычислительный центр им. А. А. Дородницына РАН.

Научный руководитель:	кандидат физико-математических наук А. И. Голиков
Официальные оппоненты:	доктор физико-математических наук, профессор В. И. Цурков, кандидат физико-математических наук, М. А. Посыпкин
Ведущая организация:	Учреждение Российской академии наук Институт системного анализа РАН

Защита состоится "____"_____ 2009 в ____ ч. ____ мин. на заседании Диссертационного совета Д 002.017.02 при Учреждении Российской академии наук Вычислительный центр им. А. А. Дородницына РАН по адресу: 119991, г. Москва, ул. Вавилова, дом 40, тел: (499) 135-24-89.

С диссертацией можно ознакомиться в библиотеке ВЦ РАН.

Автореферат разослан "____"_____ 2009.

Учёный секретарь Диссертационного совета
доктор физико-математических наук Рязанов В. В.

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Объект исследования и актуальность темы. В связи с прогрессом науки и техники возрастает объем обрабатываемой информации, при этом часто возникает необходимость решать оптимизационные задачи все большей размерности. Особенно эта тенденция заметна в таких областях, как биоинженерия и нанотехнология. Для решения задач большой размерности наряду с увеличением вычислительной мощности компьютеров необходима разработка новых алгоритмов и методов, эффективно использующих вычислительные ресурсы и позволяющих получить решения для ранее нерешаемых или труднорешаемых задач.

Существенные затруднения связаны с высокой размерностью решаемых задач. Многие прикладные задачи экономики и управления могут быть представлены в виде задач линейного программирования (ЛП), для решения которых давно существуют численные методы и программное обеспечение, и, как могло бы показаться, не представляет никакого труда найти оптимальное решение. Однако размерности современных задач (миллионы переменных и сотни тысяч ограничений) порою не позволяют решить их традиционными способами, поэтому требуется разработка новых подходов решения с привлечением мощных вычислительных ресурсов. Наличие больших, более мощных и доступных мультипроцессорных компьютеров означает, что параллельное программирование является важным и перспективным направлением современной вычислительной науки, которое целесообразно применять для решения оптимизационных задач большой размерности.

Другие затруднения могут быть связаны с отсутствием эффективных методов решения NP-трудных задач математического программирования. Это класс задач, для которых характерны такие отличительные признаки, как экспоненциальный рост вычислительной сложности. Часто попытки решения NP-трудных задач дискретной или глобальной оптимизации большой размерности связаны с использованием тех или иных эвристических методов.

В 1975г. Дж. Холландом было показано, что поиск решения оптимизационной задачи можно представить в виде эволюции группы решений, что привело к развитию генетического алгоритма (ГА) как достаточно эффективной техники оптимизации, которая моделирует процесс естественной эволюции, открытый Ч. Дарвином. Тем не менее остается актуальной разработка генетических алгоритмов, учитывающих специфику конкретной оптимизационной задачи в максимальной степени. Примером таких задач являются важная с практической точки зрения задача раскроя материалов и задача покрытия множества системой его

подмножеств, к которой сводятся многие практические задачи.

Задачи нахождения глобального оптимума являются сложными с вычислительной точки зрения, поэтому весьма актуальной является разработка эффективного метода решения конкретной практической задачи. В настоящее время доступны для решения за приемлемое время на современных компьютерах задачи с сотнями переменных. Примером такой практической задачи глобальной оптимизации, которая одновременно является и известным тестом для проверки эффективности предлагаемого метода, есть задача нахождения глобального экстремума поверхности потенциальной энергии атомного кластера.

В связи с вышеизложенным, **целью** диссертационной работы является разработка эффективных методов и их программная реализация для решения больших оптимизационных задач на примере задачи линейного программирования, раскроя материалов, покрытия множества его подмножествами и задачи глобальной оптимизации структуры атомного кластера с использованием точных и генетических алгоритмов.

В соответствии с целью исследования были поставлены и выполнены следующие конкретные задачи:

1. Разработка методов решения задачи линейного программирования с большим числом переменных и ограничений, программная реализация и проведение вычислительных экспериментов;
2. Разработка параллельных схем обобщенного алгоритма Ньютона решения больших задач линейного программирования и их программная реализация;
3. Разработка алгоритма решения задачи одномерного раскроя листовых материалов различных типоразмеров на основе генетического алгоритма и алгоритма генерации столбцов;
4. Исследование задачи покрытия множества и разработка алгоритма ее решения на основе генетического алгоритма;
5. Разработка и реализация эффективных генетических алгоритмов для поиска глобального минимума потенциальной энергии атомного кластера.

К **методам исследования**, примененным в данной работе, относятся: теория и численные методы оптимизации, линейная алгебра, параллельные методы программирования.

Научная новизна:

1. Разработан новый эффективный метод решения двойственной задачи ЛП с большим числом переменных и небольшим числом

ограничений. Метод основан на новой вспомогательной выпуклой кусочно-квадратичной функции с числом переменных, равным числу ограничений в задаче ЛП, и применении для ее безусловной минимизации глобального конечносходящегося обобщенного метода Ньютона. Показана возможность получения проекции точки на множество решений двойственной задачи ЛП, начиная с некоторого порогового значения коэффициента, входящего во вспомогательную функцию. Найдена оценка этого порогового значения коэффициента;

2. Предложены и программно реализованы параллельные схемы обобщенного метода Ньютона для вспомогательных задач безусловной оптимизации при решении задач линейного программирования;

3. Предложен комбинированный алгоритм решения задачи одномерного раскроя листовых материалов различных размеров с использованием генетического алгоритма и метода генерации столбцов;

4. Представлен новый алгоритм нахождения минимального покрытия множества на основе генетического алгоритма, проведено сравнение полученных результатов с имеющимися результатами в известной базе данных, показавшее высокую эффективность разработанного алгоритма;

5. Разработан генетический алгоритм оптимизации структуры атомных кластеров. С его помощью найдены новые решения, отсутствующие в известной Кембриджской базе данных.

Практическая ценность работы состоит в том, что разработаны, исследованы и программно реализованы методы для решения больших задач оптимизации: линейного программирования, раскроя листовых материалов, задачи о покрытии множества и задачи оптимизации структуры атомного кластера. Разработанный алгоритм решения задачи одномерного раскроя листовых материалов различных типоразмеров внедрен и успешно используется в производственном процессе трубопрокатного завода VG PIPE (Вьетнам).

Результаты, выносимые на защиту, состоят в следующем:

1. Разработаны, программно реализованы генетические алгоритмы решения следующих задач:

   - одномерного раскроя листовых материалов разных размеров;
   - задачи о покрытии множества системой его подмножеств;
   - глобальной оптимизации структуры атомных кластеров;

2. Разработан, теоретически исследован и программно реализован конечный метод нахождения проекции заданной точки на множество решений двойственной задачи ЛП с большим числом переменных;

3. Получена оценка порогового значения параметра, начиная с которого метод находит проекцию заданной точки на множество решений двойственной задачи ЛП;

4. Предложены и программно реализованы параллельные схемы обобщенного метода Ньютона для вспомогательной задачи безусловной оптимизации при решении задач линейного программирования.

Апробация работы. Результаты работы докладывались и обсуждались на 13-й Всероссийской конференции "Математическое программирование и приложения"(Екатеринбург, Россия, 2007 г.), на 9-м Международном семинаре по компьютерным наукам и информационным технологиям CSIT'07 (Уфа, Россия, 2007 г.), на XIV Байкальской международной школе-семинаре (Иркутск, Байкал, Россия, 2008 г.), а так же на семинарах отдела прикладных проблем оптимизации ВЦ РАН (Москва, Россия, 2006-2008 гг.).

Публикации. По теме диссертации опубликовано шесть печатных работ.

Объем и структура работы. Диссертация состоит из введения, трех глав, заключения и списка литературы. Общий объем работы составляет 116 страниц и список литературы из 115 наименований.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Во **введении** обоснована актуальность исследуемой проблемы, сформулирована цель и задачи диссертационной работы, перечислены полученные в диссертации новые результаты, их практическая ценность, представлены положения, выносимые на защиту и описана структура диссертации.

В **первой главе** рассматривается генетический подход для решения больших задач оптимизации. В **§1.1** дано общее описание генетического алгоритма как метода решения сложных оптимизационных задач. Далее рассмотрены две NP-трудные задачи раскроя листовых материалов и покрытия множества, а также задача глобальной оптимизации структуры атомного кластера.

В **§1.2** приведена модель задачи одномерного линейного раскроя материала, впервые предложенная Л.В. Канторовичем, в виде задачи целочисленного программирования с неявно заданной информацией о

столбцах матрицы раскроя $A = (a_{ij})$:

$$1DCSP_G(m, L, l, b) = \min \sum_{j=1}^{n} x_j = \min \|x\|, \tag{1}$$

$$\sum_{j=1}^{n} a_{ij} x_j \geq b_i, \quad i = 1, ..., m, \tag{2}$$

$$\sum_{i=1}^{m} l_i a_{ij} \leq L, \quad j = 1, ..., n, \tag{3}$$

$$x_j \in \mathbb{Z}_+, \quad j = 1, ..., n. \tag{4}$$

В этой задаче x_j - целочисленные переменные, обозначающие интенсивность применения шаблона раскроя j в решении, n - количество шаблонов, L – ширина листа раскраиваемого материала, m – количество различных заготовок, раскраиваемых из исходного листового материала, l_i - ширина заготовки типа i, а b_i есть требуемое количество заготовок этого типа, т.е. заказ на этот тип продукт.

На практике чаще встречаются задачи раскроя с различными исходными материалами, являющиеся расширением классической задачи раскроя листов одинакового размера. Ниже представлена модель для этой расширенной задачи.

Предположим, что $a^j,\ j = 1, \ldots, n$, - все шаблоны раскроя и каждый элемент x_j вектора $x^\top = (x_1...x_n)$ показывает количество применений шаблона a^j. Заметим, что n может быть очень большим. Таким образом, получим следующую модель:

$$1DMCSP(m, M, l, b, L) = \min \sum_{j=1}^{n} \hat{c}_j x_j, \tag{5}$$

$$\sum_{j=1}^{n} a_{ij} x_j \geq b_i, \quad i = 1, ..., m, \tag{6}$$

$$\sum_{i=1}^{m} l_i a_{ij} \leq \sum_{i=1}^{M} L_i a_{(i+m)j}, \tag{7}$$

$$\sum_{i=1}^{M} a_{(i+m)j} = 1, \tag{8}$$

$$x \in \mathbb{Z}_+^n, \quad a^j \in Z_+^{m'}, \quad j = 1, ..., n, \tag{9}$$

где $m' = m + M$, $L^\top = (L_1...L_M)$ - вектор ширины листов материала. Вектор $a^\top = (a_1...a_{m'}) \in \mathbb{Z}_+^{m'}$ представляет один шаблон раскроя, если выполнены ограничения (7)-(8). Элемент $a_i,\ i = 1, \ldots, m$, определяет, сколько заготовок вида i будет раскроено. Элемент a_{k+m} равен единице, если используется материал вида k, а остальные элементы $a_{i+m}, i \in \{1, ..., M\} \setminus k$, равны нулю. Если шаблон раскроя a^j использует материал вида k, то $\hat{c}_j = c_k$, где c_k - элемент вектора стоимости листов материалов.

Пусть x_* – оптимальное решение задачи (5)-(9). Введем обозначения:

- $\Omega(k)$, $k = 1, ..., M$ – множество всех шаблонов раскроя листа материала вида k, соответствующее решению x_* задачи (5)-(9);
- $x_*/\Omega(k)$ – проекция x_* на $\Omega(k)$;
- b^k – вектор заготовок, полученный из (5)-(9) в соответствии с элементами $x_*/\Omega(k)$.

Теорема 1.1. *Если x_* – оптимальное решение задачи $1DMCSP(m, M, l, b, L)$ (5) – (9), то $x_*/\Omega(k)$ есть оптимальное решение задачи $1DCSP_G(m, L_k, l, b^k)$ (1) – (4), $k = 1, \ldots, M$.*

На основе теоремы 1.1 можно представить задачу $1DMCSP$ в следующем виде:

$$1DMCSP(m, M, l, b, L) = \min \sum_{k=1}^{M} 1DCSP_G(m, L_k, l, b^k)c_k, \tag{10}$$

$$\sum_{k=1}^{M} b^k \geq b, \tag{11}$$

$$b^k \in \mathbb{Z}_+^m. \tag{12}$$

Каждый набор векторов b^k, удовлетворяющих $\sum_{k=1}^{M} b^k = b$, назовем разбиением вектора b.

Теорема 1.2. *Оптимальное решение задачи* (10) – (12) *определено на множестве разбиений вектора b.*

Теорема 1.2 позволяет ограничить пространство поиска решения задачи $1DMCSP$ внутри множества всех разбиений вектора заказа. Теоремы 1.1 и 1.2, а также новое представление задачи $1DMCSP$ в виде (10)-(12) наводят на идею разбиения задачи $1DMCSP$ на базисные задачи $1DCSP_G(m, L_k, l, b^k)$ для последующего применения метода генерации столбцов. Оптимальные решения всех базисных задач будут собраны для получения допустимого решения задачи $1DMCSP$. Поиск оптимального решения задачи $1DMCSP$ проводится генетическим алгоритмом над пространством поиска, являющимся множеством всех разбиений вектора заказа b.

Алгоритм GA-CG

Шаг 0. Создать популяцию из H особей $G_0 = \{s_{10}, ..., s_{H0}\}$. Инициализация начальной популяции легко проводится путем создания H разбиений вектора b, каждое разбиение соответствует одной особи

популяции. Особи представлены следующим образом:

$$s_{i0} = (b^1_{i0}, ..., b^M_{i0}),\ b^j_{i0} \in \mathbb{Z}^m_+,\ \sum_{j=1}^{M} b^j_{i0} = b, i = 1, ..., H.$$

Шаг 1. Решить задачу $1DCSP_G(m, L_k, l, b^k_{i\ t})$, $i = 1, \ldots, H, k = 1, \ldots, M$, где t - номер итерации (номер популяции), методом генерации столбцов. Рассчитать степени приспособленности $f(s_{i\ t})$ для каждой особи из G_t по формуле

$$f(s_{i\ t}) = \frac{1}{\sum\limits_{k=1}^{M} 1DCSP_G(m, L_k, l, b^k_{i\ t})c_k}.$$

Шаг 2. Выбрать родительские особи из G_t в зависимости от степени приспособленности для скрещивания. Пусть $s_1 = (b^1_1, ..., b^M_1)$ и $s_2 = (b^1_2, ..., b^M_2)$ - две произвольные особи. Тогда с помощью случайного числа $k, 1 < k < m$, из этих особей производим два потомка:

$$s'_1 = (\hat{b}^1_1, ..., \hat{b}^M_1),\ s'_2 = (\hat{b}^1_2, ..., \hat{b}^M_2),$$
$$\hat{b}^i_1[r] = b^i_1[r],\ \hat{b}^{i\ \top}_2[r] = b^i_2[r],\ r = 1, ..., k,\ i = 1, ..., M,$$
$$\hat{b}^i_1[j] = b^i_2[j],\ \hat{b}^i_2[j] = b^i_1[j],\ j = k+1, ..., m,\ i = 1, ..., M.$$

Получить множество особей потомков $\hat{G}_t$.

Шаг 3. Воздействовать оператором мутации на $G_t \cup \hat{G}_t$ для получения $\tilde{G}_t$. Оператор мутации воздействует на особи $s = (b^1, ..., b^M)$ для получения новой особи $s_1 = (b^1_1, ..., b^M_1)$ со случайно выбранной тройкой целых чисел (p, q, r), таких что $1 \le p \le M, 1 \le q \le M, 1 \le r \le m$, следующим образом:

$$b^i_1 = b^i \text{ при } i \ne p \text{ и } i \ne q \quad i = 1, ..., M,$$
$$b^p_1[j] = b^p[j],\ b^q_1[j] = b^q[j] \text{ при } j \ne r \text{ и } j = 1, ..., m,$$
$$b^p_1[r] = b^p[r] - 1,\ b^q_1[r] = b^q[r] + 1, \text{ если } b^p[r] > 0,$$
$$b^p_1[r] = b^p[r],\ b^q_1[r] = b^q[r], \text{ если } b^p[r] = 0.$$

Шаг 4. Совершить расчет, как на шаге 1, для особей из $\tilde{G}_t$.

Шаг 5. Вероятность отбора определяется пропорционально степени приспособленности

$$p_s = \frac{1/f(s)}{\sum_{s_i \in G} 1/f(s_i)}, \tag{13}$$

где s есть особь из рассматриваемой популяции G. Применить оператора отбора (13) над $\tilde{G}_t$ для отбора H особей с наибольшей приспособленностью, составляющих новую популяцию G_{t+1}.

Шаг 6. Если не выполнен критерий останова, то вернуться к шагу 2.

Иначе алгоритм останавливается и дает решение вместе с соответствующей матрицей раскроя.

Работа разработанного алгоритма продемонстрирована на ряде тестовых примеров. Алгоритм и его программная реализация внедрены и используются на трубопрокатном заводе VG PIPE (Вьетнам) более двух лет. С оборотом производства в 70000 тонн/год применение данного алгоритма позволило заводу сэкономить каждый год более 1000 тонн материалов.

В **§1.3** предложен эвристический алгоритм решения задачи нахождения минимального покрытия с неоднородной стоимостью, постановка которой формально определена следующим образом. Пусть имеются конечное множество $S = (\sigma_1, \sigma_2, ..., \sigma_m)$ и система его подмножеств $S_j \subset S$, $j = 1, 2, ..., n$, такие что

$$\bigcup_{j=1}^{n} S_j = S.$$

Каждому из подмножеств S_j поставлен в соответствие вес (стоимость) $c_j > 0$, таким образом рассматривается задача о взвешенном покрытии. Требуется найти минимальный по числу подмножеств набор S_j, такой что каждый элемент множества S принадлежит хотя бы одному из подмножеств этого набора. Введем матрицу $A = (a_{ij})_{m \times n}$ следующим образом:

$$a_{ij} = \begin{cases} 1, & \text{если } \sigma_i \in S_j, \\ 0, & \text{если } \sigma_i \notin S_j. \end{cases}$$

Предполагается, что каждый элемент σ_i входит хотя бы в одно из подмножеств S_j . Введем булевы переменные $x_j, j = 1, 2, ..., n$:

$$x_j = \begin{cases} 1, & \text{если } S_j \text{ входит в покрытие,} \\ 0, & \text{если } S_j \text{ не входит в покрытие.} \end{cases}$$

Тогда задача о покрытии множества состоит в нахождении покрытия с минимальной стоимостью по следующей математической модели:

$$\sum_{j=1}^{n} c_j x_j \to min, \tag{14}$$

$$\sum_{j=1}^{n} a_{ij} x_j \geq 1, i = 1, 2, ..., m, \tag{15}$$

$$x_j \in \{0, 1\}, j = 1, 2, ..., n. \tag{16}$$

Для решения задачи нахождения минимального покрытия (14)-(16) предлагается следующий генетический алгоритм:

Шаг 1. Создание начальной популяции из N случайных особей. Каждая особь k представлена n-мерным булевым вектором x^k , у которого элемент x_j^k принимает значение единицы, если подмножество S_j входит в покрытие, и принимает значение нуля, если иначе. Степень приспособленности f_k особи x^k рассчитывается следующим образом:

$$f_k = \sum_{j=1}^{n} c_j x_j^k \ .$$

Шаг 2. Выбрать две особи x^{P_1} и x^{P_2} из популяции методом пропорционального выбора (tournament selection method), используя вероятность выбора, определенную по формуле

$$p_k = \frac{1/f_k}{\sum\limits_{j=1}^{N} 1/f_j} \ . \tag{17}$$

Шаг 3. Применить оператор кроссовер над особями x^{P_1} и x^{P_2} для создания новой особи x^{Ch}. Пусть $f_{x^{P_1}}$ и $f_{x^{P_2}}$ есть степени приспособленности родительских особей x^{P_1} и x^{P_2} соответственно и определяются по (17), x^{Ch} есть потомок, полученный в результате применения оператора кроссовера над x^{P_1} и x^{P_2}. Так как гены принимают только значения из набора $\{0, 1\}$, обозначим через $p_0(j)$ частоту появления значения нуля и через $p_1(j)$ частоту появления значения единицы для j-го гена в популяции. Тогда для всех j, $1 \le j < n$, мы определим x_j^{Ch} по следующему правилу:

1. Если $x_j^{P_1} = x_j^{P_2}$, присвоим $x_j^{Ch} = x_j^{P_1} = x_j^{P_2}$.
2. Если $x_j^{P_1} \neq x_j^{P_2}$, то
   (а) при $x_j^{P_1} = 0$: присвоим $x_j^{Ch} = x_j^{P_1}$, если $f_{x^{P_1}} \cdot p_0(j) \ge f_{x^{P_2}} \cdot p_1(j)$, иначе присвоим $x_j^{Ch} = x_j^{P_2}$;
   (б) при $x_j^{P_1} = 1$: присвоим $x_j^{Ch} = x_j^{P_1}$, если $f_{x^{P_1}} \cdot p_1(j) \ge f_{x^{P_2}} \cdot p_0(j)$, иначе присвоим $x_j^{Ch} = x_j^{P_2}$.

Шаг 4. Воздействовать над x^{Ch} оператором мутации. Вместо использования классической постоянной частоты мутации применяется переменная частота мутации. Для каждого j-го гена, $1 \le j \le n$, рассчитываем соответствующую энтропию

$$H_j = -p_0(j) \log p_0(j) - p_1(j) \log p_1(j),$$

где $p_0(j)$ и $p_1(j)$ - определенные выше частоты появления значения 0 и 1 в j-м гене в популяции. Определим вероятность мутации j-го гена, $1 \le j \le n$,

по следующей формуле:

$$p_{mutation}(j) = \frac{1/H_j}{\sum\limits_{j'=1}^{n} 1/H_{j'}}.$$

Оператор мутации будет применен над потомками, созданными оператором кроссовер. Гены, имеющие вероятность мутации больше $1/n$, будут выбраны для мутации по следующему правилу замены битов:

1.Если $p_{mutation}(j) > 1/n, x_j^{Ch} = 1, p_1(j) > p_0(j),$ то $x_j^{Ch} = 0.$
2.Если $p_{mutation}(j) > 1/n, x_j^{Ch} = 0, p_0(j) > p_1(j),$ то $x_j^{Ch} = 1.$

Шаг 5. Применить алгоритм восстановления допустимости решения на x^{Ch} для получения допустимой и неизбыточной особи $x^{Ch'}$. Если $x^{Ch'}$ совпадает с какой-либо особью в популяции, то вернуться к шагу 2. Иначе перейти к следующему шагу.
Шаг 6. Заменить на $x^{Ch'}$ случайно выбранную особь со степенью приспособленности выше среднего (средней степени приспособленности популяции) из популяции.
Шаг 7. Повторить шаги 2-6 до того момента, когда больше не рождаются новые особи в популяции. Тогда решением задачи является особь с наименьшей степенью приспособленности в популяции.

Работоспособность разработанного алгоритма проверена на тестовых задачах библиотеки OR-library, доступной в Интернете (http://people.brunel.ac.uk/ mastjjb/jeb/info.html). Результаты вычисления показывают, что для задач средних размеров ($m = 200 - 400$, $n = 1000 - 4000$), для которых известно оптимальное решение, алгоритм дает оптимальное решение в большей части экспериментов и среднее отклонение не превышает 1.12%. Для 8 из 20 задач больших размерностей классов E-H ($m = 500 - 1000$, $n = 5000 - 10000$) алгоритм дает лучшее решение, чем представленные в библиотеке OR-library. Среднее время расчетов составляет от несколько секунд для первой группы задач и достигает несколько минут для второй группы.

Задача оптимизация структуры атомного кластера подробно рассмотрена в **§1.4**. Дан обзор некоторых точных и эвристических методов поиска глобального минимума. Представлен новый генетический алгоритм нахождения множества достаточно хороших конфигураций атомных кластеров с точки зрения минимальной потенциальной энергии.
Шаг 1. Инициализация начальной популяции
Для получения возможных структур кластеров из N атомов используется известная оптимальная структура кластера меньшей размерности $(N - q), q \geq 1$ в качестве “зародыша”. Остальные q атомов получаются

путем раздвоения случайно выбранного атома из $(N - q)$ и разнесения их на расстояния r_{min}. Так же производится проверка расстояния между всеми парами атомов $r(i, j)$ в кластере на предмет удовлетворения условия $r(i, j) \geq r_{min}$, где r_{min} есть минимальное межатомное расстояние для рассматриваемого вида атомного кластера. Если атомы удалены меньше, чем на такое расстояние, то кластер будет иметь высокую энергию и таким образом не будет иметь физического смысла. После получения начальных кластеров-особей, каждый из них обязательно подвергается процессу локальной оптимизации методом сопряженного градиента. Значения энергии этих локально оптимальных структур кластера будут использованы в качестве меры приспособленности особей.

Шаг 2. Кроссовер (скрещивания)

Отбор пар родителей для скрещивания производится турнирным способом. Особи-родители делятся плоскостью, перпендикулярной случайной оси и проходящей через центр масс кластера, на две “половинки”. Эти “половинки” затем обмениваются для получения двух потомков. Особи-потомки подвергаются локальной оптимизации.

Шаг 3. Мутация

Вероятность мутации определяется динамически и зависит от среднего значения энергии всех особей в популяции. Чем меньше отклонение средней энергии популяции нового поколения от предшествующего, тем выше вероятность мутации. В соответствии с полученной вероятностью выбираются особи из популяции для операции мутации. Выбранные особи "режутся"на две части случайной плоскостью перпендикулярно некоторой оси, одна из них поворачивается на случайной угол φ. Особи, полученные вследствие мутации, тоже подвергаются локальной оптимизации.

Шаг 4. Отбор новой популяции

Отбор особей для перехода в следующее поколение происходит по принципу естественного отбора. Из всех особей текущей популяции, а также потомков и мутантов, отбираются лучшие N_{pop} особи, которые составят новое поколение.

Шаг 5. Критерии останова

Если количество поколений больше заданной величины $MaxGen$ или отклонение средней энергии особей популяции меньше заданной малой величины tol, то алгоритм останавливается, иначе перейти к шагу 2.

Для численной проверки работы предложенного алгоритма использовались два модельных кластера – Ленарда-Джонса и Морса. Потенциальная энергия взаимодействия атомов в кластере Ленарда-Джонса как функция от расстояния r между двух атомов определяется

формулой:

$$v^{LJ}(r) = \frac{1}{r^{12}} - \frac{2}{r^6},$$

а потенциальная энергия Морса определена следующим образом:

$$v_\rho^M(r) = [e^{\rho(1-r)} - 1]^2 - 1.$$

Заметим, что в функции энергии Морса присутствует параметр ρ, с ростом которого увеличивается сложность поиска глобального экстремума, обычно рассматриваются $\rho = 3, 6, 10, 14$.

Проведено сравнение работы предложенного алгоритма с алгоритмом монотонного случайного спуска по локальным минимумам (Monotonic Basin Hopping) для некоторых кластеров Ленарда-Джонса, которое показало эффективность предложенного алгоритма с точки зрения меньшего количества вызовов функции локальной оптимизации почти в 2 раза.

Параллельный вариант предложенного алгоритма был реализован на C++ с использованием MPI. Для кластеров Морса с $N_{atom} \leq 80$ алгоритм показал возможность нахождения значений оптимумов, совпадающих с представленными в Кэмбриджской базе данных, за приемлемое время. Ниже в таблице представлены некоторые результаты работы алгоритма на комплексе MVS-6000 для некоторых кластеров Морса с $N_{atom} > 80$. Размер популяции во всех задачах был установлен 30. Значение f_*^M есть найденное значение потенциальной энергии Морса, P - количество использованных процессоров, а время расчета - время работы алгоритма до нахождения f_*^M. Отметим, что значение оптимальной потенциальной энергии Морса при $N_{atom} > 80$ пока неизвестно.

N_{atom}	ρ	f_*^M	P	время расчетов, мин.
85	3	-748.643598	20	5
85	6	-405.026458	20	12
85	10	-368.440724	50	16
85	14	-363.531055	50	164
90	3	-807.445287	40	5
90	6	-433.355380	40	11
90	10	-393.04228	80	24
90	14	-388.401652	80	178
95	3	-867.268837	80	6
95	6	-459.671925	80	14

В **главе 2** предлагается метод решения двойственной задачи ЛП с большим числом неотрицательных переменных и средним числом ограничений-равенств. В **§2.1** рассматривается задача нахождения проекции заданной точки на расширенное множество решений двойственной задачи и дополнительных переменных.

Пусть задана прямая и двойственная задачи ЛП соответственно в следующем виде:

$$f_* = \min_{x \in X} \; c^\top x, \quad X = \{x \in R^n \; : \; Ax = b, \; x \geq 0_n\}. \qquad (P)$$

$$f_* = \max_{u \in U} \; b^\top u, \quad U = \{u \in R^m : \; A^\top u \leq c\}. \qquad (D')$$

Здесь $A \in R^{m \times n}$, $c \in R^n$ и $b \in R^m$ заданы, x – вектор прямых переменных, а u – двойственных, через 0_i обозначен i-мерный нулевой вектор.

Предположим, что множество решений X_* прямой задачи (P) непусто, следовательно, множество решений U_* двойственной задачи (D') также непусто.

Всюду ниже двойственную задачу (D') будем представлять в следующем эквивалентном виде:

$$f_* = \max_{u \in U} \; b^\top u, \quad W = \{u \in R^m, \; v \in R^n \; : \; A^\top u + v = c, \; v \geq 0_n\}, \qquad (D)$$

где v - вектор неотрицательных дополнительных переменных. Через $W_* = [U_* \times V_*]$ обозначим множество решений задачи (D).

Из множества решений W_* двойственной задачи (D) выделим решение $\hat{w}_* = [\hat{u}_*, \hat{v}_*]$, ближайшее в евклидовой норме к некоторому заданному вектору $\hat{w} = [\hat{u}, \hat{v}]$, т.е. найдем единственное решение $\hat{w}_* = [\hat{u}_*, \hat{v}_*]$ задачи строго выпуклого квадратичного программирования

$$\frac{1}{2}(\|\hat{u}_* - \hat{u}\|^2 + \|\hat{v}_* - \hat{v}\|^2) = \min_{w=[u,v] \in W_*} \; \frac{1}{2}(\|u - \hat{u}\|^2 + \|v - \hat{v}\|^2), \qquad (18)$$

$$W_* = \{u \in R^m \; , v \in R^n_+ \; : \; A^\top u + v = c, \; b^\top u = f_*\}.$$

Здесь f_* – оптимальное значение целевой функции исходной задачи ЛП.

Двойственная задача к (18) является задачей безусловной максимизации

$$\max_{y \in R^n} \; \max_{\alpha \in R^1} \{-c^\top y - \hat{u}^\top(\alpha b - Ay) - \tfrac{1}{2}\|(\alpha b - Ay)\|^2 + \alpha f_* + \tfrac{1}{2}\|\hat{v}\|^2 - \\ -\tfrac{1}{2}\|(\hat{v} - y)_+\|^2\}. \qquad (19)$$

К сожалению, задача безусловной оптимизации (19) содержит неизвестную априори величину f_* – оптимальное значение целевой функции задачи ЛП. Однако эту задачу можно упростить, избавившись от этого недостатка. Для этого вместо (19) предлагается решать следующую упрощенную задачу безусловной максимизации:

$$I = \max_{y \in R^n} \; S(y, \alpha, \hat{w}), \qquad (20)$$

где скаляр α фиксирован, а функция $S(y, \alpha, \hat{w})$ определена следующим образом:

$$S(y, \alpha, \hat{w}) = -c^\top y + \hat{u}^\top A y - \frac{1}{2}\|\alpha b - Ay\|^2 - \frac{1}{2}\|(\hat{v} - y)_+\|^2.$$

Без потери общности предположим, что первые l компонент вектора $\hat{v}_*$ строго больше нуля. Тогда векторы $\hat{v}_*$, $\hat{v}$, y и матрица A представимы в виде

$$\hat{v}_*^\top = [\hat{v}_*^{l\top}, \hat{v}_*^{d\top}],\ \hat{v}^\top = [\hat{v}^{l\top}, \hat{v}^{d\top}],\ y^\top = [y^{l\top}, y^{d\top}],\ A = [A_l \mid A_d],$$

где $\hat{v}_*^l > 0_l$, $\hat{v}_*^d = 0_d$, $d = n - l$.

Определим величину α_* следующим образом:

$$\alpha_* = \inf_{\alpha \in R^1} \inf_{y^d \in R^d} \{\ \alpha\ :\ \alpha b - A_d y^d = \hat{u}_* - \hat{u} - A_l(\hat{v}_*^l - \hat{v}^l),\ y^d \geq \hat{v}^d\}. \quad (21)$$

Показано, что ограничения в (21) совместны, но целевая функция может быть не ограничена снизу. В этом случае полагается, что $\alpha_* = \gamma$, где γ– некоторое число. Если система уравнений в (21) однозначно разрешима относительно y^d, то α_* представима в виде

$$\alpha_* = \begin{cases} \max\limits_{i \in \sigma} \frac{(\hat{v}^d + (A_d^\top A_d)^{-1} A_d^\top (\hat{u}_* - \hat{u} - A_l(\hat{v}_*^l - \hat{v}^l))^i}{(x_*^d)^i}, & \text{если } \sigma \neq \emptyset. \\ \gamma > -\infty, \qquad \text{если } \sigma = \emptyset. \end{cases} \quad (22)$$

Здесь введено индексное множество $\sigma = \{l + 1 \leq i \leq n\ :\ (x_*^d)^i > 0\}$ и γ — произвольное число.

Теорема 2.1 *Пусть множество решений W_* исходной задачи ЛП непусто. Тогда при любом $\alpha \geq \alpha_*$, где α_* вычисляется из* (21)*, проекция $\hat{w}_* = [\hat{u}_*, \hat{v}_*]$ заданной точки $\hat{w} = [\hat{u}, \hat{v}]$ определяется по формулам*

$$\hat{u}_* = \hat{u} + \alpha b - Ay(\alpha),$$
$$\hat{v}_* = (\hat{v} - y(\alpha))_+,$$

где $y(\alpha)$ – решение задачи безусловной оптимизации (20)*.*

Если дополнительно матрица A_d, соответствующая нулевым компонентам $\hat{v}_*$, имеет полный ранг, то α_* определяется по формуле (22)*, а точное решение прямой задачи (P) находится в результате решения задачи безусловной максимизации* (20) *по формуле*

$$x_*^d = \frac{1}{\alpha}(y^d(\alpha) + (A_d^\top A_d)^{-1} A_d^\top (\hat{u}_* - \hat{u} - A_l(\hat{v}_*^l - \hat{v}^l)). \quad (23)$$

Теорема 2.1 позволяет заменить задачу (19), содержащую априори неизвестное число f_*, на задачу (20), в которой вместо этого

числа фигурирует полуинтервал $[\alpha_*, +\infty)$, что существенно проще с вычислительной точки зрения.

Отметим, что значение α_*, найденное из решения задачи линейного программирования (21) или формулы (22), может быть отрицательным. Это означает, что для двойственной задачи (D) проекция точки $\hat{w}$ на множество ее решений W_* совпадает с проекцией этой точки на допустимое множество W.

Следующая теорема утверждает, что если известна какая-нибудь точка $w_* \in W_*$, то можно получить решение прямой задачи (P) после однократного решения задачи безусловной максимизации (20).

Теорема 2.2 *Пусть множество решений W_* задачи ЛП (D) непусто. Тогда для любых $\alpha > 0$ и $\hat{w} = w_* \in W_*$ точное решение прямой задачи (P) находится по формуле $x_* = y(\alpha)/\alpha$, где $y(\alpha)$ — решение задачи безусловной максимизации* (20).

Для одновременного решения прямой и двойственной задач ЛП можно использовать следующий итерационный процесс:

$$u_{k+1} = u_k + \alpha b - Ay_{k+1}, \tag{24}$$

$$v_{k+1} = (v_k - y_{k+1})_+, \tag{25}$$

где произвольный параметр $\alpha > 0$ фиксирован, а вектор y_{k+1} определяется из решения следующей задачи безусловной максимизации:

$$y_{k+1} \in \arg\max_{y \in \mathbb{R}^n} \{-c^\top y + u_k^\top Ay - \frac{1}{2}\|(\alpha b - Ay)\|^2 - \frac{1}{2}\|(v_k - y)_+\|^2\}. \tag{26}$$

Этот итерационный процесс является конечным и дает точное решение прямой задачи (P) и точное решение двойственной задачи (D). Отметим, что при использовании этого метода не требуется знать пороговое значение коэффициента штрафа.

Безусловная максимизация в (20) может выполняться любым методом, например методом сопряженного градиента. Но, как показал О. Мангасарьян, для безусловной оптимизации кусочно-квадратичной функции особенно эффективен обобщенный метод Ньютона.

Предлагаемый метод решения прямой и двойственной задач ЛП (метод ДП) сочетает итерационный процесс (24)-(25) и обобщенный метод Ньютона, примененный для решения задачи (26). Метод реализован в системе MATLAB в виде программы DLP в **§2.2**.

В конце расчетов вычислялись чебышевские нормы векторов невязок:

$$\Delta_1 = \|Ax - b\|_\infty, \quad \Delta_2 = |c^\top x - b^\top u|, \quad \Delta_3 = \|(A^\top u + v - c)\|_\infty.$$

Ниже в таблице приведены результаты численных расчетов по программе DLP на компьютере P-C2D, 2.4 Ghz.,1 Gb., $\hat{u}_0 = 0_m$, $\hat{v}_0 = 0_n$, $y_0 = 1_n$, ρ– плотность заполнения A ненулевыми элементами.

$m \times n \times \rho$	T, *сек.*	I *шаг*	α,	Δ_1,	Δ_2,	Δ_3,
$10^6 \times 100 \times 0.01$	3.8	2	10^{-5}	1.4×10^{-10}	4.5×10^{-7}	1.8×10^{-10}
$10^6 \times 500 \times 0.01$	18.6	2	2×10^{-5}	5.5×10^{-10}	4.3×10^{-8}	1.0×10^{-9}
$10^6 \times 700 \times 0.01$	26.7	2	8×10^{-5}	3.7×10^{-10}	1.1×10^{-7}	2.0×10^{-10}
$10^6 \times 1000 \times 0.01$	39.5	2	5×10^{-5}	6.5×10^{-11}	1.3×10^{-7}	1.0×10^{-11}
$10^6 \times 2000 \times 0.01$	80.4	2	8×10^{-5}	5.5×10^{-11}	6.2×10^{-8}	1.1×10^{-10}
$10^6 \times 3000 \times 0.01$	244.1	4	5×10^{-5}	5.6×10^{-11}	2.1×10^{-8}	4.3×10^{-11}
$10^4 \times 500 \times 1$	34.7	5	4.5×10^{-4}	1.1×10^{-10}	9.8×10^{-9}	8.9×10^{-10}
$10^4 \times 1000 \times 1$	210.7	7	0.2	5.5×10^{-12}	3.0×10^{-9}	1.0×10^{-7}
$3000 \times 2000 \times 1$	241.1	10	0.1	6.2×10^{-11}	2.4×10^{-9}	3.0×10^{-10}
$5000 \times 1000 \times 1$	96.2	10	0.01	5.5×10^{-11}	4.8×10^{-9}	5.7×10^{-11}
$10^4 \times 4000 \times 0.01$	67.7	3	0.01	2.9×10^{-12}	7.2×10^{-10}	4.9×10^{-12}
$(5.10^6) \times 500 \times 0.01$	93.6	2	5×10^{-6}	5.9×10^{-10}	1.6×10^{-7}	4.7×10^{-11}
$(5.10^6) \times 1000 \times 0.01$	198.0	3	5×10^{-6}	4.5×10^{-10}	6.1×10^{-7}	2.6×10^{-11}
$10^7 \times 500 \times 0.01$	177.2	3	10^{-6}	1.0×10^{-9}	2.4×10^{-7}	4.6×10^{-10}

В **третьей главе** предлагаются, исследуются и экспериментально проверяются четыре параллельные схемы решения задач ЛП с помощью метода, предложенного Ю. Г. Евтушенко и А. И. Голиковым. Метод был подробно исследован и реализован в системе MATLAB (программа EGM) в диссертационной работе Н. Моллаверди. Численные эксперименты и сравнение программы EGM с известными коммерческими и исследовательскими программами показали превосходство программы EGM и возможность решать задачи ЛП с 50 миллионами переменных, но с небольшим числом ограничений (не более 4 тысяч). Актуальность параллельной реализации является очевидной для увеличения числа ограничений решаемых задач.

В **§3.1** описание реализации начинается с расчетных формул рассматриваемого итерационного метода

1. Задается начальное приближение x^0 и p^0, $\tilde{p} = p^0$.

2. Вычисляется функционал

$$S(p^k, \beta, x^s) = b^\top p^k - \frac{1}{2}||(x^s + A^\top p^k - \beta c)_+||^2. \qquad (27)$$

Здесь s - номер внешней итерации, а k - номер внутренней итерации метода Ньютона для решения задачи безусловной максимизации

$$p_{k+1} \in \arg\max_{p \in R^m} \{b^\top p - \tfrac{1}{2}\|(x_k + A^\top p - \beta c)_+\|^2\}.$$

3. Вычисляется градиент функционала (27) по переменной p:

$$G^k = \frac{\partial S}{\partial p}(p^k, \beta, x^s) = b - A(x^s + A^\top p^k - \beta c)_+. \qquad (28)$$

4. Формируется обобщенная матрица Гессе функционала (27) по переменной p (здесь удобно использовать матрицу Гессе со знаком минус):
$$H^k = \delta I + AD^k A^\top. \tag{29}$$
Диагональная матрица $D^k \in \mathbb{R}^{n\times n}$ задается равенствами
$$(D^k)_{ii} = \begin{cases} 1, & \text{если} \quad (x^s + A^\top p^k - \beta c)_i > 0, \\ 0, & \text{если} \quad (x^k + A^\top p^k - \beta c)_i \le 0. \end{cases}$$
Таким образом, $H^k \in \mathbb{R}^{m\times m}$.

5. Находится направление максимизации δp как приближенное решение линейной системы
$$H^k \delta p = -G^k. \tag{30}$$

6. Находится $p^{k+1} = p^k - \tau_k \delta p$.

7. Если достигнута сходимость внутренних итераций по k, то положим $\tilde{p} = p^{k+1}$. Вычисляется x^{s+1} по формуле
$$x^{s+1} = (x^s + A^\top \tilde{p} - \beta c)_+,$$
$p^0 = \tilde{p}$ и повторяются шаги 2 - 6.

В **§3.2** рассматриваются основные операции параллельного алгоритма. Параллельная реализация алгоритма 1-7 требует распараллеливания следующих основных операций:

- умножение матрицы на вектор: Ax и $A^\top p$;
- скалярное произведение вида $x^\top y$, $x, y \in \mathbb{R}^m$ и $p^\top q$, $p, q \in \mathbb{R}^n$;
- формирование обобщенной матрицы Гессе (29);
- умножение обобщенной матрицы Гессе на вектор;

Заметим, что все оставшиеся вычисления на этапах алгоритма 1 - 7 являются локальными. Например, метод сопряженных градиентов требует вычисления распределенных скалярных произведений и распределенного умножения матрицы на вектор. Остальные вычисления локальны и не требуют обменов. При вычислении функционала (27) необходимо распределенное умножение матрицы на вектор для вычисления вектора $z^k = (x^s + A^\top p^k - \beta c)_+$ и для вычисления скалярного произведения $b^T p^k$. Для вычисления градиента (28) нужна дополнительная операция умножения матрицы на вектор.

В **§3.3** последовательно приводятся описания четырех схем разбиения данных по процессорам для распараллеливания рассматриваемого

алгоритма. Поскольку число строк m в матрице A значительно меньше, чем число столбцов n, то при небольшом числе процессоров n_p можно использовать простую столбцовую схему разбиения данных, в которой матрица A разбивается на блочные столбцы A_i примерно равного размера, как показано на рис.1. Все векторы размерности m, т.е. p, b и т.д., дублируются на каждом из процессоров. С такой схемой хранения операция умножения на транспонированную матрицу, т.е вычисление выражения $A^T p$, является локальным. С другой стороны, умножение матрицы A на вектор можно записать в виде

$$Ax = \sum_{i=1}^{n_p} A_i x_i.$$

При этом умножение подматриц на подвекторы x_i производится независимо, и n_p полученных векторов размерности m складываются друг с другом при помощи функции `MPI_Allreduce` из библиотеки MPI.

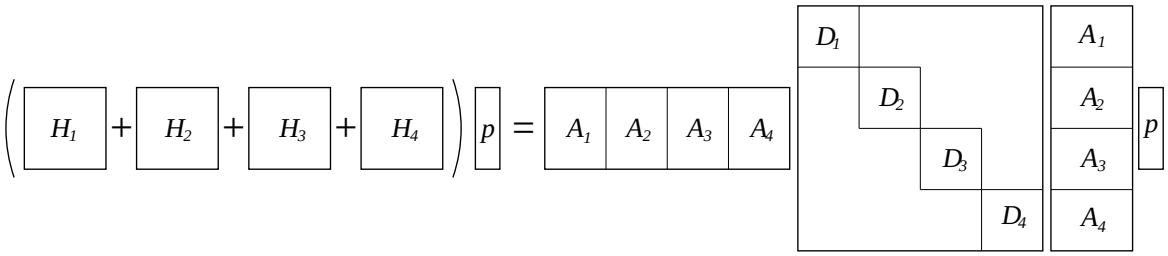


Рис. 1. Столбцовая схема: разбиение матриц и векторов

Матрицу Гессе H можно представить в виде

$$H = \sum_{i=1}^{n_p} H_i, \text{ где } H_i = A_i D_i A_i^T,$$

где $D_i \in \mathbb{R}^{N_c \times N_c}$ - диагональные блоки матрицы D. В столбцовой схеме матрица H не формируется. На каждом процессоре вычисляется локально матрица H_i. Для умножения матрицы на вектор

$$Hq = \sum_{i=1}^{n_p} H_i q$$

каждое слагаемое вычисляется локально, и n_p получившихся векторов суммируются при помощи функции `MPI_Allreduce`, так что результат суммирования оказывается доступен на всех процессорах.

Таким образом, метод сопряженных градиентов для решения линейной системы (30) не является параллельным и его вычислительные

затраты растут пропорционально числу процессоров. Ясно, что такой простой алгоритм эффективен лишь в том случае, если отношение вычислительных затрат на решение линейной системы к оставшимся вычислительным затратам на одной итерации метода Ньютона достаточно мало.

Для эффективного распараллеливания метода сопряженных градиентов предлагается использовать строчную схему разбиения данных, в которой матрица A разбивается на блочные строки A_i примерно равного размера, как показано на рис.2.

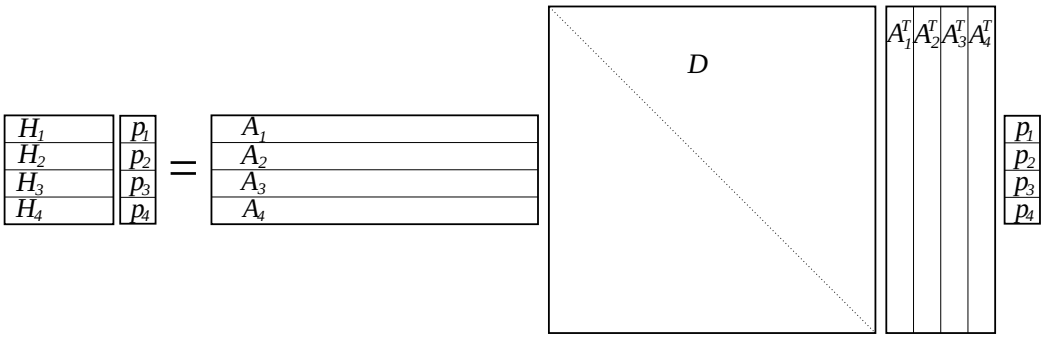


Рис. 2. Строчная схема: разбиение матриц и векторов

Однако если хранить на каждом процессоре только соответствующую подматрицу, то при формировании обобщенной матрицы Гессе возникает необходимость обмена всех подматриц между процессорам, что означает большие коммуникационные затраты. Для избежания этих обменов на каждом процессоре матрица A хранится целиком, хотя каждый процессор i отвечает только за свою подматрицу $A_i \in \mathbb{R}^{N_r \times n}$. Все векторы размерности n, т.е. x, c и т.д., дублируются на каждом из процессоров. С такой схемой хранения операция умножения матрицы A на $A^\top$ становится локальной. Эта схема хранения также позволяет сократить объем коммуникационных затрат для умножения транспонированной матрицы на вектор. После умножения матрицы $A_i^\top$ на вектор p_i необходимо обмениваться длинными векторами (размерности n), что влечет большие коммуникационные затраты. Поэтому умножение производится не блоками, а собираются вместе все части вектора p_i со всех процессоров воедино, и умножается целиком матрица $A^\top$ на вектор p. Минус этого подхода в том, что операция $A^\top p$ не распараллеливается, но так как большая часть вычислений на этапе формирования системы линейных уравнений обычно приходит на построение матрицы Гессе, то эти лишние вычисления являются целесообразными для избежания больших обменов.

Для того чтобы достичь большей параллельной эффективности, можно использовать “клеточное” разбиение, в котором матрица A

разбивается на прямоугольные блоки, как показано на рис.3.

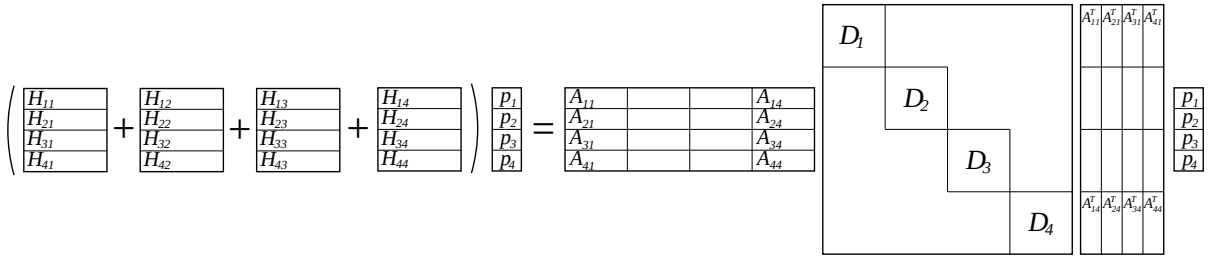


Рис. 3. Клеточная схема: разбиение матриц и векторов

Для простоты изложения предположим, что полное число процессоров можно представить в виде $n_p = n_r \times n_c$, т.е. достаточно условно можно полагать, что процессоры расположены в узлах решетки размера $n_r \times n_c$, при этом величина n делится нацело на n_c, а m делится нацело на n_r. Таким образом, $n = n_c \times N_c$ и $m = n_r \times N_r$. Таким образом, матрица A состоит из подматриц $A_{ij} \in \mathbb{R}^{N_r \times N_c}$. В этой схеме вектор $x \in \mathbb{R}^n$ разбивается на n_c подвекторов x_i, а вектор $p \in \mathbb{R}^m$ на n_r подвекторов p_j, причем подвектор x_i лежит одновременно на n_r процессорах, принадлежащих j-му столбцу "решетки", также, как и все подматрицы A_{*j}.

Основные распределенные операции для такого разбиения данных будут проводиться вначале в рамках процессоров одного столбца "решетки" по строчной схеме, а затем по всем процессорам одной строки, как в столбцовой схеме.

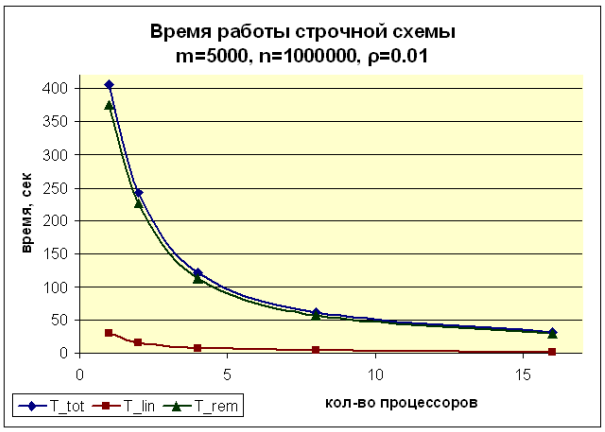
Для решения задач ЛП с большим числом переменных (порядка несколько миллионов) и большим числом ограничений (порядка несколько сотен тысяч), где критичны размерности задачи, а следовательно, и объем требуемой памяти, предлагается новая схема, наиболее экономичная по использованию памяти. Матрицы разбиваются так же, как показано на рис.2. Однако в этой схеме на каждом процессоре хранится только одна строчная подматрица A_i, а матрица H в явном виде не хранится.

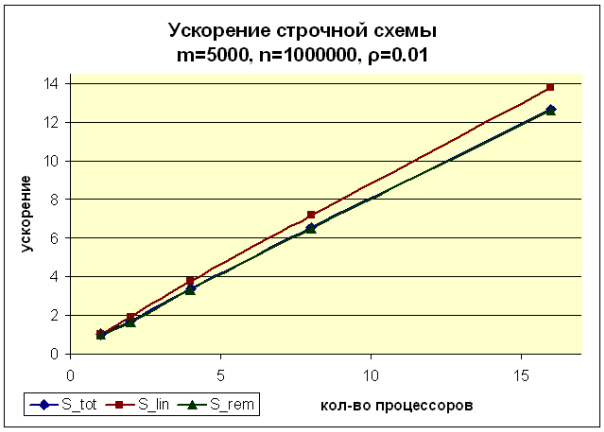
На каждой внешней итерации метода вычисляется только градиент, соответствующая подматрица Гессе H_i не вычисляется. Далее на каждой внутренней итерации (метода сопряженных градиентов) каждый процессор вычисляет произведение своей подматрицы Гессе H_i на вектор p_i путем последовательного вычисления двух произведений матрицы на векторов: $v_i = DA_i^\top p_i$ и $H_i = A_i \sum\limits_i v_i$.

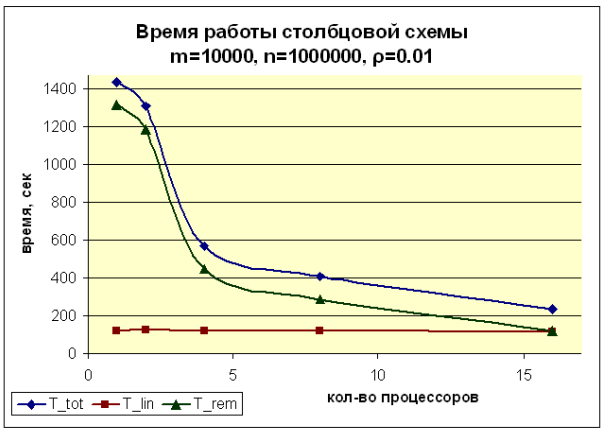
Для суммирования длинных разреженных векторов v_i по всем процессорам был предложен специальный способ суммирования разреженных векторов. Это позволило в некоторой степени уменьшить

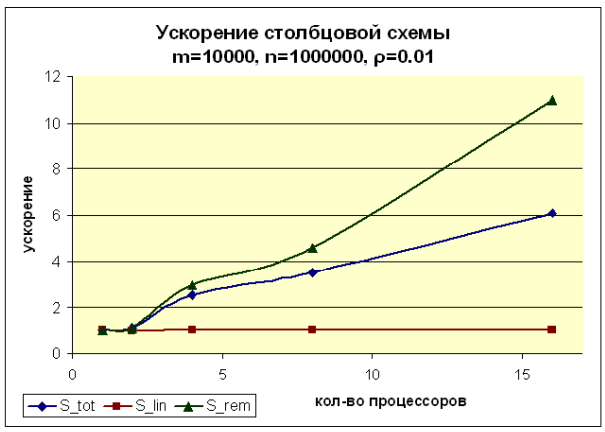
коммуникационные затраты.

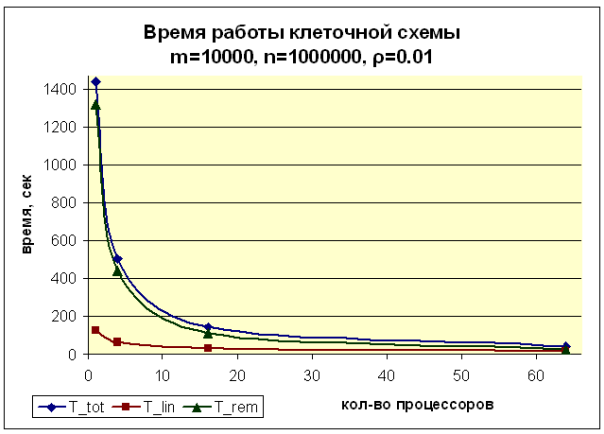
Ниже приведены графики зависимости общего времени работы алгоритма (T_{tot}), времени решения систем уравнений (T_{lin}) и времени на остальных операциях(T_{rem}), а также ускорения от числа процессоров для трех схем.

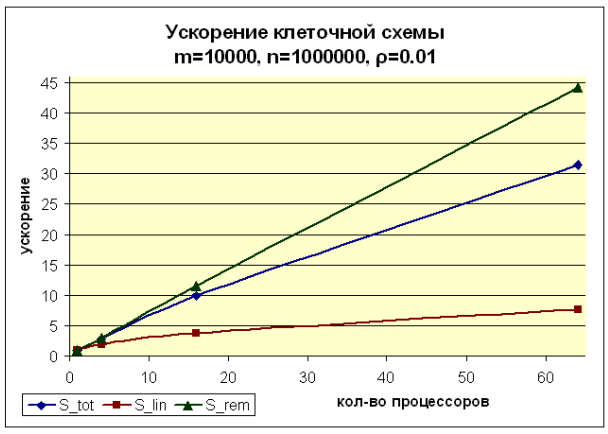












Строчная схема наиболее предпочтительна с точки зрения ускорения, но ее применение к задачам ЛП ограничено размерностью памяти на каждом процессоре. Столбцовая схема наиболее эффективна при очень большом числе переменных и среднем числе ограничений. В других случаях целесообразно применять клеточную схему. Но если число

ограничений очень велико (порядка сотен тысяч), то следует применять безматричную схему. Ниже в таблице приводятся результаты работы безматричной схемы.

$m \times n \times \rho$	n_p	T_{tot}	Δ_1	Δ_2	Δ_3
$5 \cdot 10^4 \times 10^6 \times 0.01$	16	400.02	$2.1 \cdot 10^{-6}$	$2.1 \cdot 10^{-7}$	$1.1 \cdot 10^{-10}$
$10^5 \times 10^6 \times 0.01$	20	484.62	$8.1 \cdot 10^{-6}$	$3.6 \cdot 10^{-6}$	$5.6 \cdot 10^{-11}$
$10^5 \times 2 \cdot 10^6 \times 0.01$	40	823.13	$4.5 \cdot 10^{-6}$	$4.2 \cdot 10^{-7}$	$7.2 \cdot 10^{-11}$
$2 \cdot 10^5 \times 2 \cdot 10^6 \times 0.01$	80	2317.42	$4.9 \cdot 10^{-5}$	$6.6 \cdot 10^{-6}$	$5.2 \cdot 10^{-10}$

Из таблицы видно, что безматричная схема позволяет решать задачи ЛП с большим числом ограничений (m=200 тыс., n=2 млн., 80 процессоров, время решения 40 мин.).

Основное содержание диссертационной работы изложено в следующих публикациях:

1. Nguyen Minh Hang et al. *A model combining genetic algorithm and simplex method for solving a production expense minimizing problem.* // Journal of Computer Science and Cybernetics, No.22, Vol.4, Hanoi, 2006. P.319-324 (in Vietnamese).

2. Нгуен М.Х. *Применение генетического алгоритма для решения задачи планирования производства.* // Тр. XIII-й Всеросс. конф. “Математическое программирование и приложения”, Екатеринбург, Россия, Март 2007. С. 132-133.

3. Nguyen M.H. *About one application of genetic algorithm in Production planning problem.* // Proc. of the 9th Int. Workshop on Comp. Sci. and Inf. Techn. Ufa, 2007 (Sept. 13-16). Vol.1. Pp. 225-229.

4. Нгуен М.Х. *Применение генетического алгоритма для решения одной задачи планирования производства.* // Динамика неоднородных систем. М.:Изд-во ЛКИ, 2007. Т. 29. Вып. 11. С.160-167.

5. Нгуен М.Х. *Двухэтапный метод решения одной задачи раскроя материалов.* // Тр. XIV Байкальской междунар. школы-семинара. “Приложение методов оптимизации”. Т. 4. Иркутск-Северобайкальск, Россия. Июль 2008. С.192-202.

6. Нгуен М.Х. *Применение генетического алгоритма для задачи нахождения покрытия множества.* // Динамика неоднородных систем. М.:Изд-во ЛКИ, 2008. Т. 33. Вып. 12. С.206-219.