ФЕДЕРАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЦЕНТР «ИНФОРМАТИКА И УПРАВЛЕНИЕ» РОССИЙСКОЙ АКАДЕМИИ НАУК

На правах рукописи
УДК 519.854.3

ЛЕМТЮЖНИКОВА ДАРЬЯ ВЛАДИМИРОВНА

ПОНИЖЕНИЕ РАЗМЕРНОСТИ ДЛЯ БОЛЬШИХ ЗАДАЧ С РАЗРЕЖЕННЫМИ МАТРИЦАМИ.

Специальность 05.13.17 —
«Теоретические основы информатики»

Диссертация на соискание учёной степени
кандидата физико-математических наук

Научный руководитель:
д.ф.-м.н., профессор
Цурков В.И.

Москва – 2017

Содержание

Введение

Разреженные матрицы появляются при постановке задач из многих научных и инженерных областей. Эффективные методы хранения и обработки таких матриц в современных вычислительных системах вызывают интерес у широкого круга исследователей. Одним из актуальных приложений разреженных матриц является решение соответствующих задач дискретной оптимизации (ДО). ДО является эффективным инструментом для моделирования многих практических задач. Это касается таких известных постановок: размещение объектов, планирование ресурсов, покрытие поверхностей, сетевая оптимизация, маршрутизация, логистика, теория расписаний, искусственный интеллект, анализ данных, робототехника и т. п. Выделение специальных структур в разреженных матрицах позволяют существенно сократить время решения.

Большинство интересных задач являются NP–трудными [1, 2]. Многие задачи ДО, возникающие на практике, содержат огромное число неизвестных и ограничений, поэтому они трудно решаемы.

С другой стороны модели ДО для больших практических задач часто представляют собой системы, подсистемы которых слабо связанны между собой, и таких подсистем достаточно много. Поэтому естественным подходом для решения таких задач представляется разбиение на подзадачи. В связи с этим особую актуальность приобретают декомпозиционные подходы — способы разбиения больших задач на подзадачи [3–40].

Развитие информационных технологий [41, 42], появление многопроцессорных комплексов, суперкомпьютеров создало условия для разработки алгоритмов ДО с распараллеливанием вычислений [43–71]. Поэтому разработка в задачах ДО декомпозиционных алгоритмов и исследование возможности их реализации чрезвычайно важно.

Эффективными алгоритмами для решения разреженных задач ДО являются локальные элиминационные алгоритмы (ЛЭА) [72]. ЛЭА объединяют локальные алгоритмы декомпозиции [73], алгоритмы несериального динами-

ческого программирования [74, 75], а также алгоритмы сегментной элиминации [76]. Распараллеливание вычислительного процесса локального элиминационного алгоритма [77] может существенно ускорить решение задач ДО большой размерности.

Базовыми в работе является квазиблочная структура разреженной матрицы и её обработка. С развитием вычислительной техники более актуальным становится вопрос сокращения вычислений. Для решения разреженных задач ДО естественным образом выбирается алгоритм, использующий локальные области соответствующей матрицы. В [78] приводятся первые результаты вычислительного эксперимента, где с помощью ЛЭА решались различные задачи ДО. В [79] содержатся результаты исследования эффективности локального алгоритма для решения особого класса задач ДО — квазиблочных задач. Также в [79] продемонстрировано, что асимптотическая средняя оценка эффективности локального алгоритма слабо зависит от алгоритма ДО, решающего подзадачи. Такая зависимость объясняется следующим образом: средняя оценка эффективности была исследована на множестве всех квазиблочных структур, но, как было замечено в [80], локальные алгоритмы являются эффективными для задач с ограниченной связностью блоков [81]. Эффективность локального алгоритма теоретически и экспериментально исследована недостаточно полно, поэтому остаётся актуальным поиск удачных модификаций ЛЭА и его сочетаний с различными точными и приближенными решателями ДО [72].

Автор принимал участие в разработке локально–элиминационного алгоритма, который квалифицирован как новые и актуальные направления в информатике. **Основной целью** данного исследования является повышение эффективности данного подхода, выделение класса задач, для которых применим метод, его ускорение, возможности решения задач большой размерности путём распараллеливания. Фактически речь идёт о выявлении закономерностей в больших массивах данных, в качестве которых выступают разреженные матрицы. Исследование их свойств и нахождение правил оперирования с такими матрицами также является целью диссертации.

Основные положения, выносимые на защиту.

1. Получены системы неравенств для блочно–лестничной и блочно–древовидных структур в общем виде, а также относительно нескольких классов разреженных матриц, которые устанавливают зависимость меж-

ду степенью квазиблочной структуры и числа её блоков в зависимости от размерности матрицы и числа ненулевых элементов в ней.

2. Разработаны алгоритмы выделения квазиблочной структуры для разреженных матриц.

3. В рамках теории локальных элиминационных алгоритмов введены новые понятия, а также обоснована зависимость между графовыми структурами в связи с проблемой оптимального порядка элиминации.

4. Разработаны модификации локального элиминационного алгоритма.

5. Осуществлено распараллеливание больших задач ДО с матрицей квазиблочной структуры на системе GRID.

Научная новизна. В данной работе впервые сформулированы и доказаны теоремы, устанавливающие связь между параметрами матрицы и соответствующей квазиблочной структуры. Также впервые исследованы и реализованы методы выделения квазиблочной структуры для разреженных матриц. Они используют алгоритм [82] для поиска квазиблочных структур в разреженных матрицах, который практически не использовался и автору неизвестны попытки его программной реализации. Введены понятия и доказаны свойства графовых структур, соответствующих порядку элиминации. Соответствующие теоремы дают основу доказательства важных свойств в проблеме нахождения оптимального исключения переменных. Протестировано влияние порядка элиминации на скорость ЛЭА. Впервые предложен и реализован ряд модификаций ЛЭА для разреженных задач ДО с квазиблочной структурой. Реализовано распараллеливание задач с квазиблочной стрктурой на GRID.

Научная и практическая значимость. В данной работе разработана техника понижения размерности больших разреженных матриц и соответствующих задач ДО. Исследование окрестностей переменных, определение декомпозиции задач с квазиблочной структурой, модификации локального элиминационного алгоритма и его распараллеливание продолжают ряд исследований Ю.И. Журавлёва, Ю.Ю. Финкельштейна, В.И. Цуркова, О.А. Щербины.

Разработанные методы позволяют получить решение задачи ДО большой размерности при невозможности получить её решение за приемлемое время.

Теоретические результаты диссертационной работы вошли в состав курса «Дискретная оптимизация», читаемого студентам 4–го курса на кафедре «Интеллектуальные системы» ФУПМ МФТИ.

Связь с плановыми научными исследованиями. Работа выполнена в рамках грантов Российского фонда фундаментальных исследований № 16–51–53093 «Разработка эффективных алгоритмов решения специальных задач оптимизации и приложений» и № 16–51–55019 «Метод обобщенной разреженной оптимизации для распознавания сложных ригидных объектов на изображениях и в видеопотоке».

Степень достоверности полученных результатов подтверждается проработкой литературных источников по теме диссертации, реализованной постановкой необходимого количества численных расчётов, а также современной методикой исследования, которые соответствуют поставленным в работе целям и задачам. Научные положения, выводы и рекомендации, сформулированные в диссертации, подкреплены убедительными фактическими данными, наглядно представленными в приведенных таблицах и рисунках. Подготовка полученных результатов проведена с использованием современных программных средств.

Апробация работы. Основные результаты работы докладывались на конференциях:

- X Международная конференция "Интеллектуальные системы и компьютерные науки"(Москва, 5–10 декабря 2011 г.).
- V Международная конференция "Танаевские чтения"(Минск, 28–29 марта 2012 г.).
- XVIII Международная конференция "Ломоносов 2012"(Москва, 9–13 апреля 2012 г.).
- IV Международная конференция "Problems of Cybernetics and Informatics (PCI)"(Баку, 12–14 сентября 2012 г.)
- V Всероссийская конференция "Проблемы оптимизации и экономические приложения"(Омск, 02–06 июля 2012 г.).
- VI Международная конференция "Distributed Computing and Grid–technologies in Science and Education"(Europe/Moscow, 30 июня – 5 июля 2014);

- 17—ая Всероссийская конференция (Светлогорск, 19–23 сентября, 2015)
- 18—ая Всероссийская конференция (Таганрог, 9–13 октября, 2017)

Также результаты были изложены на семинарах: в Сколково, на мехмате МГУ, в ФИЦ ИУ РАН, ИППИ РАН и др.

Личный вклад. Автор составил обзор по разреженным матрицам, исследовал их особенности и сформулировал ряд теорем, устанавливающих связь между матрицей и соответствующей квазиблочной структурой. Были исследованы алгоритм выделения квазиблочной структуры, предложил и реализовал его модификации. Автор составил обзор по декомпозиционным методам, а также сформулировал ряд понятий и доказал свойства графовых структур, соответствующих порядку элиминации и протестировано его влияние на скорость локального элиминационного алгоритма. Автором были разработаны модификации локального элиминационного алгоритма, осуществлена параллельная модификация ЛЭА была выполнена на GRID.

Публикации. Основные результаты по теме диссертации изложены в 20 печатных изданиях [77, 83–101], 9 из которых изданы в журналах, рекомендованных ВАК [77, 90, 93, 95, 97–100], 11 — в тезисах докладов [83–87, 89, 91, 92, 94, 96, 101].

Объем и структура работы. Диссертация состоит из введения, трёх глав, заключения и одного приложения. Полный объем диссертации составляет 180 страниц с 33 рисунками и 6 таблицами. Список литературы содержит 264 наименования.

Глава 1

Квазиблочная структура в разреженных матрицах и связь её параметров

В данной главе рассматриваются разреженные матрицы, для которых выделяются квазиблочные структуры, а именно — блочно–древовидные и блочно–лестничные. Формулируется ряд теорем, в которых устанавливается связь между компонентами квазиблочной структуры в зависимости от размерности матрицы и числа ненулевых элементов в ней. Приводятся алгоритмы для выделения квазиблочных структур.

1.1 Разреженные матрицы большой размерности

Разреженные матрицы встречаются в таких областях, как математическое моделирование, теория управления, структурный анализ и др. Матрица A является разреженной, если содержит преимущественно нулевые элементы. Интерес к разреженности заключается в экономии на вычислениях. Здесь естественным образом возникает проблема хранения разреженных матриц. Для хранения используются разные, например, ленточные. Способы представления делятся следующим образом. Выделяют полные и неполные схемы в зависимости от того, представлена вся матрица или только её часть. Также выделяют упорядоченные и неупорядоченные схемы в зависимости от того, произволен порядок хранения элементов или упорядочен. Далее рассмотрим наиболее распространенные способы представления разреженных матриц [102]. Достаточно известным является координатный формат, в котором хранятся ненулевые элементы матрицы, номера их строк и столбцов. Данный метод применим к произвольной матрице, которая хранится в массиве ненулевых элементов, массиве номеров строк и массиве номеров столбцов матрицы. Такое представление является полным,

потому что представлена матрица целиком, и неупорядоченным, потому что порядок хранения элементов произволен. Ещё один широко используемый метод хранения — разреженный строчный формат, состоящий из массива ненулевых элементов, массива номеров столбцов и массива указателей позиций p: с них начинается описание следующей строки. Массив ненулевых элементов упорядочен: элементы перечислены по строкам, начиная с первой. В массивах ненулевых элементов и номеров столбцов хранится описание k–й строки в позициях с $p[k]$–й по $(p[k+1]-1)$–ю. Причём k–я строка пустая в случае, когда $p[k] = p[k+1]$. Для матрицы, состоящей из n строк, длина массива указателей позиций — $n+1$. Аналогичным способом строится разреженный столбцовый формат относительно столбцов. Эти два формата удобны для реализации основных операций, таких как: перестановка строк и столбцов, сложение и умножение матриц, нахождение обратной матрицы, транспонирование и т. п. Кроме того, данные способы представления являются полными и упорядоченными, поскольку элементы каждой строки (столбца) хранятся в соответствии с возрастанием индексов столбцов (строк). Симметричную разреженную матрицу можно хранить в качестве треугольной подматрицы. Если большая часть элементов, располагающихся по диагонали — ненулевые, для их хранения достаточно отдельного массива.

Для базовых операций с разреженными матрицами, а именно — умножение матрицы на вектор, транспонирование матрицы, умножение матрицы на матрицу, — существуют специальные алгоритмы. Также для многих алгоритмов необходимо определять матрицы перестановки. Ряд алгоритмов направлен на упорядочивание элементов в разреженных матрицах. Локальные стратегии обработки разреженных матриц (например, алгоритм минимальной степени) заключаются в упорядочивании разреженной матрицы. Также используется упорядочивание матриц для получения специальных форм (например, метод рекурсивного разбиения).

Основой для всех алгоритмов, которые предназначены для решения систем уравнений с квадратной матрицей, является метод последовательного исключения неизвестных (гауссово исключение, метод Гаусса). Рассмотрим основные модификации метода Гаусса [102]. Метод LU–разложения заключается в представлении матрицы в виде произведения в виде двух матриц L и U, где L — нижняя треугольная матрица, диагональные элементы которой равны еди-

нице, а U — верхняя треугольная матрица, диагональные элементы которой — нули. Метод Холецкого используется для решения систем уравнений, соответствующая матрица которых действительна, симметрична и положительно определена. Он основан на представлении исходной матрицы в произведение L и L^T, где L — нижняя треугольная матрица, элементы на главной диагонали которой положительны. Метод прогонки (алгоритм Томаса) предназначен для ленточных матриц, то есть матриц, все ненулевые элементы которых находятся вблизи главной диагонали. Рассмотрим подход на примере трёхдиагональной матрицы. Метод прогонки основывается на предположении, что искомые неизвестные связаны рекуррентным соотношением: $x_i = \alpha_{i+1}x_{i+1} + \beta_{i+1}$, где $i = n-1, n-2, \ldots, 1$. С помощью этого соотношения можно выразить переменные x_{i-1}, x_i и x_{i+1} и подставить в исходное уравнение $A_i x_{i-1} + C_i x_i + B_i x_{i+1}$. Отсюда можно выразить прогоночные коэффициенты α_i и β_i и получить решение системы. Метод редукции применим для матриц размера, равного степени двойки. Его идея заключается в последовательном исключении из системы $a_i x_{i-1} + c_i x_i + b_i x_{i+1} = f_i, 1 \leq i \leq n-1$, где $x_0 = 0, x_n = 0$, неизвестных сначала с нечетными номерами, затем с номерами, кратными 2 (но не кратными 4), и т.д., и восстановлении значений нечетных переменных на основании известных значений переменных с четными номерами.

Отметим также алгоритм Кроута, алгоритм Дулитла и QR–разложение, описанные в [103–108]. Они связаны с различными прямыми методами решения алгебраических систем линейных уравнений, представленных разреженными матрицами. Все эти методы алгебраически эквивалентны с незначительным различием в последовательности вычислений.

В случае больших разреженных систем линейных уравнений предпочтение отдаётся итерационным методам, поскольку они не приводят к появлению на итерациях новых ненулевых элементов и оказываются более эффективными по затратам машинного времени [109, 110]. Наиболее известные из итерационных методов — методы простой итерации, Якоби, Гаусса–Зейделя, последовательной верхней релаксации, симметричной последовательной верхней релаксации. Например, метод Якоби эффективен для ленточных матриц. В случае пятидиагональной матрицы матрица представляется в виде пяти векторов. Для вычисления компоненты вектора решения необходимо выполнить четыре операции умножения и сложения и одну операцию деления. Метод сопряженных гради-

ентов применим для решения систем линейных уравнений, соответствующая разреженная матрица которых симметрична и положительно определена. Данный подход является одним из методов Крылова [109], строящих ортогональный базис подпространства $span\{r_0, Ar_0, A^2r_0, \ldots, A^ir_0\}$ для некоторого стартового вектора r_0 . Решение исходной системы ищется на этом подпространстве путем минимизации невязки. Эффективность итерационных методов для решения систем линейных уравнений с разреженными матрицами показана в работах [111, 112].

Далее рассмотрим численные алгоритмы для выделения треугольной формы. Эта форма позволяет рассматривать набор линейных уравнений как последовательность подзадач. Здесь можно выделить два основных подхода: поиск поперечных перестановок, заключающийся в перестановках элементов на диагонали (например, алгоритм поиска в глубину [109]), использование симметричных перестановок (например, алгоритм Тарьяна [113] и алгоритм Сарджента–Уэстерберга [114]).

С. Писсанецки [115] рассматривает технику применения разреженных матриц для различных алгебраических задач. В [116] изучается задача управления линейной динамической системой, в которой связи между подсистемами являются слабыми. Это характеризуется разреженными матрицами перекрестных взаимодействий. В указанных матрицах лишь незначительное число коэффициентов отличны от нуля. Предлагаемый метод сначала приводит задачу к каноническому виду, затем идентифицирует переменные в подсистемах, которые сильно взаимодействуют друг с другом. Это осуществляется с помощью введения так называемой матрицы порогового уровня. Ее коэффициенты формируются из собственных значений матриц подсистем и из матриц, входящих в функционал исходной задачи. Строится субоптимальное управление, которое учитывает сильные и игнорирует слабые связи между подсистемами.

В книге А. Джорджа и Дж. Лю [110] описаны основные методы решения разреженных положительно определенных линейных систем, а также излагаются алгоритмы параллельных и вложенных сечений, предназначенные для систем метода конечных элементов. Вычисления с разреженными матрицами стали основой построения и исследования хордальных графов [117–119]. В [120] рассматривается уровень дробления разреженных задач для их эффективного решения с помощью параллельных схем. Для работы с разреженными матрицами необ-

ходимо использовать специальные алгоритмы и структуры данных, которые учитывают структуру матрицы. Основные программные библиотеки, созданные для работы с разреженными матрицами, это SparseLib++ [1] и uBLAS [2] для C++, SPARSPAK [3] для Фортрана, CSparse [4] для Си, а также Модуль Sparse из библиотеки SciPy (Python)[5]. Среди новейших исследований относительно разреженных матриц представляются интересными следующие работы.

Во многих задачах обработки изображения и компьютерного зрения данные имеют форму матриц. Традиционные методы часто вытягивают матрицу в вектор (столбец) и используют подходы для векторов. Эти методы игнорируют положение элементов матрицы, а преобразованный вектор часто имеет очень большую размерность. Вопрос о том, как напрямую выбирать признаки для двумерной матрицы непосредственно, до сих пор является важной открытой задачей. В статье [121] предлагается алгоритм регрессии разреженных матриц (sparse matrix regression, SMR) для прямого выбора признаков в матричных данных. В этом алгоритме используется модель регрессии матриц, которая принимает на вход матрицу и отображает каждую матрицу на её метку. Используя внутренние свойств коэффициентов регрессии, авторы разделяют разреженные ограничения на коэффициенты, чтобы сформировать вектор признаков. Предложен эффективный метод оптимизации с доказанным свойством сходимости. Показано, что число векторов регрессии можно рассматривать в качестве параметра баланса между способностью к обучению и обобщающей способностью. Чтобы показать эффективность алгоритма SMR, авторы сравнили его с несколькими алгоритмами на основе вытягивания в столбец на нескольких наборах тестовых данных. Кроме того, авторы оценили качество работы SMR в задаче классификации сцен.

Умножение разреженных матриц обычно производится в оперативной памяти, а масштабирование на случаи больших размерностей осуществляется при помощи использования распределенной памяти на множестве узлов. В отличие от традиционного подхода в статье [122] умножение разреженных матриц масштабируется за пределы емкости памяти в случае умножения разреженной матрицы на плотную (sparse matrix dense matrix multiplication, SpMM) при по-

[1] http://faculty.cse.tamu.edu/davis/welcome.html
[2] http://www.boost.org/doc/libs/1_50_0/libs/numeric/ublas/doc/index.htm
[3] http://www.netlib.org/sparspak/
[4] https://github.com/wo80/CSparse.NET
[5] https://sourceforge.net/projects/scipy/files/scipy/

мощи подхода с частично внешней памятью (semi-external memory, SEM), т.е. когда разреженная матрица помещается на типовые твердотельные накопители (SSD), а плотная матрица — в оперативной памяти. Данный подход умножения SEM-SpMM включает оптимизации в работе с памятью для больших графов со степенным распределением степеней вершин. Он превосходит такие подходы с использованием оперативной памяти, как Trilinos или Intel MKL, и хорошо распространяется на графы с миллиардами узлов, что намного больше размеров оперативной памяти. Более того, на одиночной машине с параллельными вычислениями, этот метод работает настолько же быстро как распределенные вычисления при помощи Trilinos, использовавшие в пять раз больше вычислительной мощности. Также авторами исследована реализация метода для случая оперативной памяти (IM-SpMM) для того, чтобы оценить издержки в случае хранения данных на твердотельных накопителях. Метод SEM-SpMM достигает почти 100% производительности метода только с оперативной памятью IM-SpMM для графов, когда в разреженной матрице более четырех столбцов. В общем случае метод с частично внешней памятью, SEM-SpMM, достигает 65% производительности IM-SpMM, для матриц общего вида. Авторы применили подход SpMM к трем важнейшим задачам анализа данных — PageRank, собственным разложениям и неотрицательным матричным разложениям, и показали, что подход с частично внешней памятью SEM существенно повысил достижимую производительность в решении данного класса задач.

Системы автоматической рекомендации представляют собой подкласс систем фильтрации информации, которые предсказывают различные предпочтения пользователя или предпочтение, которое пользователи отдают некоторому товару. Одна из наиболее частых проблем в таких системах — это отсутствие данных. Такая ситуация порождает сильно разреженную матрицу, что снижает точность предсказания. Особенно это критично в случае холодного старта, когда в системе появляется новый пользователь или новый товар. В [123] сделана попытка ослабить проблемы «холодного пользователя» или «холодного товара» уменьшая степень разреженности матрицы при помощи локального итеративного метода наименьших квадратов и комбинации алгоритма распределения тепла с алгоритмом распределения вероятности.

Перекрестные данные собираются из нескольких источников или образуются из–за возникновения нескольких точек зрения на одни и те же предметы.

Поскольку такая информация часто взаимодополняется или консолидируется, перекрестный анализ может дать существенное улучшение для процесса принятия решений. Особенной трудностью перекрестного анализа является вопрос о том, как эффективно исследовать сильно коррелированные многомерные данные. Методы понижения размерности предлагают эффективное решение данной задачи, однако, вопросы выбора правильной модели и параметров для понижения размерности остаются открытыми. В работе [124] предлагается эффективный алгоритм обучения на разреженных данных для понижения размерности в случае перекрестных данных. Отличительным свойством данного алгоритма является то, что он непараметрический и автоматический. В частности, авторы представляют корреляцию перекрестных данных при помощи матрицы ковариации. Затем они раскладывают эту матрицу в виде последовательности матриц меньшего ранга при помощи решения оптимизационной задачи по аналогии с методом чередующихся наименьших квадратов. Наиболее важным элементом подхода является разработанная новая непараметрическая функция, поощряющая повышение разреженности, которая позволила построить экономную модель расчетов. Проведены обширные вычислительные эксперименты на реальных данных, показывающие эффективность предложенного алгоритма. Результаты экспериментов показывают, что предложенный метод успешно конкурирует с современными алгоритмами обучения для разреженных данных.

Неотрицательные матричные разложения (NMF, HMP) являются классическими методами понижения размерности. В последнее время возник интерес к HMP в следствие обнаруженной способности решать сложные задачи извлечения данных и машинного обучения, особенно в приложении к задаче генетического анализа. Обзорная статья [125] нацелена на исследование приложений HMP в поиске дифференциально выраженных генов и кластеризации образцов. Рассматриваются основные HMP модели, их свойства, принципы и алгоритмы и их различные обобщения, расширения и модификации. Экспериментальные результаты показывают уровень производительности различных алгоритмов HMP в задачах идентификации дифференциально выраженных генов и кластеризации образцов.

Задача разбиения сетки в параллельном методе конечных элементов является NP-трудной. За несколько последних десятилетий были предложены

несколько эвристических подходов для решения этой задачи. Разработан ряд эффективных методов решения уравнений, которые используют специфические свойства больших матриц (например, симметричность и положительную определенность). В [126] обсуждается производительность распределенных методов конечных элементов, использующих различные методы определения структуры сетки (выбор разбиения) и решатели уравнений. В данной работе классифицируются методы определения разбиения сетки, исследуются различные вариации методов решения линейных и нелинейных уравнений, а также изучается влияние разбиения сетки и решателя уравнений на производительность распределенных методов конечных элементов.

В следствие высокой вычислительной сложности матричных вычислений важным становится эффективное выполнение в распределенной среде. В работе [127] предлагается подход для распределения матричных арифметических операций с разреженными матрицами по вычислительным кластерам с целью ускорения обработки матриц большой размерности. Данный подход направлен на разбиение матричных операций на независимые подзадачи с использованием существенных характеристик каждого типа арифметических операций, а также конкретных видов матриц. Подход применялся к наиболее часто используемым матричным операциям. Производительность предложенного подхода оценивалась для задачи выбора признаков большой размерности из текста, и двух наборов данных из практических задач. Экспериментальные сравнения показывают, что предложенный подход позволяет существенно уменьшить время операций над матрицами большой размерности по сравнению с последовательными и многопоточными реализациями, а также алгоритмами из рассматриваемых библиотек линейной алгебры.

В работе [128] предлагается оптимизация вычислений для широко известного алгоритма умножения разреженных матриц и векторов SpMV для процессоров Intel Xeon Phi. Архитектурные отличия этих процессоров от традиционных многоядерных процессоров обнажают существенные неотъемлемые структурные слабости операций с разреженными матрицами, увеличивая критическое влияние на производительность других факторов помимо традиционно известного ограничения по полосе пропускания памяти. В статье показано, что для таких процессоров существенно важной является адаптивность матриц. Для этого предлагается подход, который сначала определяет узкие места в мат-

ричных алгоритмах, используя или профилирование расчетов или структуру матриц, а затем выбирается конкретная оптимизация, позволяющая использовать их и преодолеть неэффективность. Набор оптимизационных алгоритмов использует широко применяемый формат хранения матриц при помощи сжатых разреженных строк (Compressed Sparse Row, CSR) и имеет низкие накладные расходы для вычислений, что делает данный подход практически применимым даже для итеративных решателей, которые сходятся за небольшое число итераций. Данный алгоритм оценен на сопроцессоре Intel Knights Corner и показано, что возможно обнаружить и соответствующим образом оптимизировать SpMV для большинства матриц на различных типах тестов, что дает существенный выигрыш производительности по сравнению с соответствующими реализациями CSR в свежих версиях библиотеки Intel MKL.

Численное моделирование физических явлений для городов является задачей высокой вычислительной сложности. В [129] рассматривается задача симулирования обмена излучением в масштабе города для разных типов городов. На основе того, что матричное представление видимости между зданиями является сильно разреженным, предлагается новая вычислительная модель расчета излучения. Матрица плотности потока излучаемой энергии, характеризующая модели размерностью до 140 тыс. поверхностей, может храниться в оперативной памяти. Предлагаемый метод оценки обратной матрицы потока излучения ускоряет расчет обмена излучением. За счёт этого учитываются характеристики окружения при проектировании зданий в оценке нормативов строительства в городских условиях.

В [130] представлена система Sympiler, которая представляет собой предметно–ориентированный генератор кода, оптимизирующий вычисления с разреженными матрицами за счет разделения фазы символьного анализа и фазы численных операций с разреженными данными. Характерные схемы в вычислительных алгоритмах для разреженных матриц определяются структурой разреженных входов и самим алгоритмом обработки разреженных данных. Во многих реальных симуляциях структура разреженных входных данных меняется незначительно или не меняется вовсе. Sympiler использует это свойство чтобы аналитически проанализировать разреженные алгоритмы на этапе компиляции и применить полученные на этапе анализа преобразования, что позволит применить низкоуровневые преобразования к алгоритмам с разреженными данными.

В результате код, генерируемый системой Sympiler, превосходит глубоко оптимизированные алгоритмы матричного разложения из известных специальных библиотек, что дает выигрыш по сравнению с Eigen и CHOLMOD в 3.8 и 1.5 раза соответственно.

Операция умножения разреженной матрицы на плотный вектор (matrix by a dense vector, SpMV) является центральным элементом многих научных вычислений: она используется в итеративных методах решения линейных систем и задачах на собственные значения для разреженных матриц. Появление графических процессоров с вычислениями общего назначения (General Purpose GPU) придало новый импульс этому направлению и появилось много новых статей посвященной данной задаче. Например, в [131] приводится обзор возможных методов реализации библиотеки операций SpMV на GPGPU, которые появились за последние несколько лет. Обсуждаются проблемы и компромиссы, с которыми столкнулись многие исследователи, и приводится список возможных решений, собранный по категориям на основе общих признаков. Также приводится сравнение производительности при использовании различных моделей GPGPU для ряда тестовых матриц из различных предметных областей.

Исследования в области человеко–машинного взаимодействия связаны с задачами сегментации, выделения целевого объекта и отслеживания. Значительный уровень интереса к данной области связан с пониманием, что многие приложения, такие как наблюдение, человеко-машинное взаимодействие, получат большой импульс развития при наличии мощных и эффективных результатов. В [132] предлагается архитектура для задачи отслеживания объекта с использованием разреженным матриц и классификатора Adaboost. Этот подход состоит из трех шагов: сначала извлекаются признаки из изображения, затем признаки представляются как разреженная матрица, а после этого используется классификатор Adaboost для корректной классификации значений из разреженной матрицы, что позволяет получить решение задачи отслеживания. Приводятся результаты экспериментов, которые показывают, что данная архитектура дает улучшение производительности по сравнению с другими подходами к задаче отслеживания объектов.

Модели на основе выделения представлений малого ранга имеют большой потенциал для задачи определения выделяющихся, броских объектов, в которых матрица разлагается в матрицу малого ранга, представляющую фон и раз-

реженную матрицу, представляющую броские, заметные объекты. Тем не менее, существуют два недостатка такого подхода. Первый заключается в том, что обычно предполагается, что элементы в разреженной матрице являются взаимно независимыми, игнорируются пространственные отношения между областями изображения или наличие устойчивых шаблонов. Второй недостаток заключается в том, что когда матрица малого ранга и разреженная матрица относительно когерентны, т.е. когда имеется сходство между заметными объектами и фоном, или когда фон имеет сложную структуру, то для известных моделей сложно их разделить. В [133] предлагается модель структурной декомпозиции матриц с двумя регуляризациями структуры:

1. Регуляризация на основе древовидных моделей, повышающая разреженность, что позволяет зафиксировать структуру изображения и усиливает степень сходства фрагментов одного и того же объекта, придавая им одинаковые значения заметности.

2. Регуляризация Лапласа, которая увеличивает зазор между броскими объектами и фоном в пространстве признаков.

Затем добавляются априорные значения высокого уровня, чтобы направлять декомпозицию матриц и ещё больше усилить выделение объектов. Модель для выделения заметных объектов оценивается на пяти сложных наборах данных, включающих изображения с одним объектом, несколькими объектами и сложные сцены. Представлены результаты в сравнении с 24 современными методами по семи различным метрикам.

Умножение матрицы на матрицу является базовой операцией линейной алгебры и существенным элементом большого количества алгоритмов в различных областях науки. Теория и реализации хорошо известны для случая плотных, квадратных матриц. Если же матрицы разреженные, с характерными для своих приложений шаблонами разреженности, то оптимальная реализация операции умножения остается открытым вопросом. В [134] исследуется влияние коммуникации (обмен информацией между агентами) на производительность 2.5D алгоритмов и влияние односторонней MPI коммуникации в контексте теории линейных масштабируемых структур электронов. В частности, авторы расширили библиотеку работы с разреженными матрицами DBSCR, которая является основным элементом для теории линейных масштабируемых структур

электронов, и слабо масштабируемых коррелированных методов. Эта библиотека специально создана для эффективного выполнения умножений матрицы на матрицу, для блочно–разреженных матриц, имеющих достаточно большое наполнение. Также сравнивается производительность исходной версии на алгоритме Кэннона при коммуникации через MPI в топологии точка–точка и алгоритма с односторонними MPI коммуникациями с удаленным доступом к памяти (RMA), как в случае 2D подхода, так и в случае 2.5D. Подход 2.5D увеличивает потребление памяти и объем дополнительных вычислений, но снижает объемы коммуникации, что может привести к повышению производительности, если коммуникации являются узким местом. Кроме того, алгоритм 2.5D легче реализовать в случае односторонних коммуникаций. Приведено подробное описание реализации алгоритмов, а также описание случая неидеальных топологий процессоров, поскольку это может быть важно для практических приложений. Вследствие важности знания точной структуры разреженности, и даже для фактических матричных данных, когда эффективное заполнение решается при умножении, все тесты проводятся в пакете CP2K с тестами типов приложений. Результаты показывают существенное увеличение производительности 2.5D алгоритма с удаленным доступом к памяти, до 1.8 раза, и обнаружено, что преимущество увеличивается с ростом числа процессов, занятых в распараллеливании.

В [135] рассматривается задача декомпозиции матрицы на матрицу малого ранга и плотную матрицу в области больших данных. Обычные алгоритмы для разложении матриц используют все данные из матрицы, чтобы получить матрицу малого ранга и плотную матрицу, и основаны на оптимизационных задачах, сложность которых возрастает с повышением размерности данных, что ограничивает их масштабируемость. Более того, известные рандомизированные подходы основываются на использовании равномерного распределения, что оказывается крайне неэффективным для многих матриц из практических задач, в которых есть дополнительная структура (например, кластеры). В данной работе рассматривается масштабируемый подход поиска подпространств (subspace–pursuit), который преобразует задачу разложения в задачу обучаемого поиска подпространств. Декомпозиция проводится с использованием небольшого фрагмента данных, образованного из выбранных столбцов и/или строк матрицы. Показано, что даже если фрагмент выбирать случайно равномерным образом,

достаточное число столбцов/строк приблизительно O(rμ), где μ является параметром когерентности, а r — ранг малоразмерной компоненты. Кроме того, для решения проблемы выбора столбцов/строк из структурированных данных, предложены адаптивные алгоритмы выбора. Приводится анализ предложенного метода, где показано, что он делает число выбираемых столбцов/строк инвариантным по отношению к распределению входных данных. Предложенный алгоритм может быть модифицирован для итеративной реализации, предложена онлайн-схема.

Ручная сегментация повреждений от ишемического инсульта в МРТ изображениях отнимает много времени и подвержена вариациям в зависимости от оценивающего. Имеется высокий интерес к надежной автоматической сегментации при клинических испытаниях и исследованиях. Однако, актуальные проблемы сегментации показывают, что даже самые современные подходы имеют недостаточную точность, и проблема сегментации повреждений является очень сложной. Использование матричной декомпозиции на матрицу малого ранга и плотную матрицу даёт ценную априорную информацию для сегментации в этой области. В статье [136] изучается применимость RPCA или подходов выделения на основе RPCA для сегментации ишемических инсультов в наборах данных МРТ изображений FLAIR. На основе наиболее перспективного метода сегментации оценивается производительность метода, использующего информацию из разреженной компоненты RPCA изображения как признак для алгоритма «случайный лес» (random forest), известного в области машинного обучения. Исчерпывающее сравнение по каждому из признаков для производительности сегментации показывают потенциальный выигрыш от использования разложения на матрицы малого ранга и разреженную матрицу в задачах выделения признаков ишемического инсульта.

При решении линейных систем при помощи итеративных методов возникает дилемма между выбором простых, но малоэффективных итераций по разреженным направлениям поиска (такими, как координатный спуск) или затратными итерациями в правильно выбранном направлении (такими, как метод сопряженных градиентов). В [137] предлагается среднее решение, а также показывается, что можно выполнять простые итерации по плотным направлениям поиска, при условии, что эти направления можно выделить на основе нового вида разреженного разложения. Например, если направление поиска —

это столбцы иерархической матрицы, тогда стоимость каждой операции имеет логарифмическую зависимость от числа переменных. Используя некоторые результаты теории графов по остовным деревьям (low-stretch spanning tree), авторы получают в частном случае имеющий почти линейную сложность по времени приближенный алгоритм нахождения решения с минимальной нормой для линейной системы $Bx = b$, где B это матрица инцидентности графа.

Разделение фона и переднего плана является начальным шагом при определении движения объектов в системах видеонаблюдения. Недавние исследования показывают, что удобной архитектурой для решения задачи отделения фона и движущихся объектов являются подходы на основе декомпозиции на матрицу меньшего ранга и разреженную матрицу. Наиболее характерным примером таких задач является устойчивый метод главных компонент (Robust Principal Component Analysis, RPCA) решаемый при помощи Principal Component Pursuit (PCP). Однако, аналогичные устойчивые явные или неявные разложения встречаются в следующих постановках задач: Robust Non-negative Matrix Factorization (RNMF), Robust Matrix Completion (RMC), Robust Subspace Recovery (RSR), Robust Subspace Tracking (RST) и Robust Low-Rank Minimization (RLRM). Основная цель этих схожих задач заключается в получении явного или неявного разложения на матрицу малого ранга и аддитивные матрицы. Эти формулировки отличаются явной или неявной декомпозицией, функцией штрафа, оптимизационной задачей и алгоритмами её решения. Задача в исходном виде может оказаться NP–трудной, при этом она может быть выпуклой или невыпуклой в зависимости от ограничений или функции потерь. В задаче выделения переднего плана от фона при формализации задачи следует принимать во внимание ограничения, характерные для фона и переднего плана, а также пространственные и временные особенности. На практике последовательность изображений фона моделируется как подпространство небольшой размерности, которое может постепенно изменяться во времени, а движущиеся объекты на переднем плане представляют собой коррелированные разреженные отклонения. На сегодняшний день не найдены такие методы, которые бы одновременно решали все проблемы, сопровождающие реальные видеопотоки, так как отсутствует четкая количественная процедура оценки на синтетических и реальных наборах данных, где имелась бы точная проверочная информация и при этом воспроизводился бы весь диапазон основных трудностей, присутству-

ющих в реальности. В статье [138] приводится обзор для аналогичной проблемы в постановках устойчивого обучения, которые основаны на декомпозиции на матрицу малого ранга и аддитивную матрицу, для сравнения существующих алгоритмов разделения переднего плана и фона. Также приводится обзор новых достижений в различных постановках задач, приведенных выше, который позволяет сформировать универсальный подход, названный разложением на матрицы малого ранга и аддитивные (Decomposition into Low-rank plus Additive Matrices, DLAM). После этого авторы анализируют каждый метод в каждой постановке устойчивого обучения, а также декомпозицию, функцию потерь, оптимизационную задачу и методы решения, которые применяются в этой постановке. Затем изучается возможность получения итеративного варианта алгоритма или варианта для работы в режиме реального времени для задачи разделения переднего плана и фона. В конце статьи приводятся результаты сравнения 32 устойчивых алгоритмов обучения на тестовом наборе данных большой размерности.

В ближайшем будущем ожидается, что распространение технологий высокоскростной оптической широкоугольной записи приведет к появлению нового направления в астрономии — кино–астрономии (movie astronomy). Объемы данных от подобных наблюдений будут колоссальными, поэтому незаменимыми станут методы эффективного сжатия данных. В [139] предлагается решение на основе аппроксимации матрицами малого ранга, в которой используется разложение разреженных матриц, которое позволяет эффективно понизить размер данных при сохранении информации достаточного качества. Авторы применили один из подобных методов к видео данным полученным с прототипа Tomoe Gozen, установленного на телескопе схемы Шмидта размером 1 м в обсерватории Кисо (Япония). Полноценная сессия наблюдения с использованием Tomoe Gozen за одну ночь даёт около 30 терабайт данных. В статье показано, что данные можно сжать примерно в 10 раз без потери быстрых событий, таких как точечные вспышки и метеоры. Интенсивность точечных источников может быть восстановлена из сжатых данных. Обработка имеет достаточно высокую скорость по сравнению с ожидаемой скоростью записи в реальных сессиях записи.

Перейдём к определению разреженности. Пусть матрица $A_{N\times M} = \{a_{ij}\}$ — данная матрица, n — число столбцов, m — число строк, $\mathfrak{z}$ — число значащих

элементов в матрице, то есть элементов, которые не являются нулями. Формализуем определение разреженности в общем смысле.

Определение 1.1. *Матрица A, множество индексов которой $\Psi = M \times N$, а подмножество индексов нулевых элементов — $\Psi^0 \subset \Psi$: $|\Psi \setminus \Psi^0| << \Psi^0$, называется разреженной.*

Замечание. Данное определение носит условный характер и не используется. Далее приведём альтернативное определение разреженности, используемое на практике. Для практических задач разреженность определяется в процентном соотношении. В самом общем случае разреженной называют задачу, в матрице которой большая часть элементов — нули, то есть, больше 50%.

Перейдём к альтернативному определению разреженной матрицы, в которой большая часть элементов — нули.

Определение 1.2.

Матрица A, множество индексов которой $\Psi = M \times N$, подмножество индексов нулевых элементов $\Psi^0 \subset \Psi$: $0.5|\Psi| \leq |\Psi^0|$, называется разреженной.

Принимая во внимание, что $|\Psi|$ — число всех элементов матрицы, выразим данную величину через число столбцов матрицы n и число сток m: $|\Psi| = mn$. Обозначим $\mathfrak{z}$ — число ненулевых элементов. Тогда определение 1.2 примет следующий вид.

Определение 1.3.

Матрица A, у которой n столбцов, m строк и $\mathfrak{z}$ ненулевых элементов, таких что выполняется $0.5mn > \mathfrak{z}$, называется разреженной.

Классифицируем разреженные матрицы для дальнейшей работы. Заметим, что матрицы многих классов задач можно отнести к одной из вышеназванных групп матриц. Существенно реже встречаются задачи, матрицы которых можно отнести к двум и более группам. Поэтому важно изучить структуру матриц таких задач в зависимости от каждой группы матриц. Многие задачи имеют особенности в матричной структуре. Эти особенности позволяют рассматривать большие матрицы как последовательность матриц меньшего ранга, связанных некоторым образом между собой. То есть, многие подзадачи, соответствующие таким матрицам, могут быть решены независимо друг от друга,

Таблица 1.1: Классификация разреженных матриц

Вид матрицы	Пояснение	Декларация
Очень широкая	Число столбцов превышает число строк в два раза и более	$n \geq 2m$
Широкая	Число столбцов превышает число строк менее чем в два раза	$0.5(n+1) < m < n$
Квадратная	Число столбцов равно числу строк	$n = m$
Узкая	Число строк превышает число столбцов	$n < m$

что существенно экономит вычислительное время. В частности, такой структурой является блочно–древовидная структура (БД–структура). Далее рассмотрим блочные структуры, вершины в которых упорядочены таким образом, что структура является связной и иерархической, то есть является древовидной.

Сформулируем необходимые понятия для определения БД–структуры в матрице. Для начала введём определения связывающего столбца в матрице и взаимосвязи столбцов.

Определение 1.4. Столбец j матрицы , в котором существует хотя бы пара ненулевых элементов a_{ij} и $b_{i' \neq i,j}$ в строках i и i', будем называть связывающим. Взаимосвязью двух столбцов j_1 и j_2 в матрице называется единовременное вхождение ненулевых элементов этих столбцов в строку i.

Перейдём к определению графа взаимосвязей.

Определение 1.5. Графом взаимосвязей $G(X, E)$ матрицы A называется граф, вершины которого соответствуют номерам ненулевых элементов матрицы, а для элементов матрицы, которые находятся в одной строке, между соответствующими вершинами есть ребро.

Замечание. Подразумевается, что индексы вершин графа взаимосвязей и номера столбцов матрицы A эквивалентны.

Замечание. Ребро (i, j) графа взаимосвязей $G(X, E)$ соответствует двум ненулевым значениям в столбцах i и j матрицы ограничений $A_{N \times M}$, которые находятся в одной строке.

Определим понятие «путь в графе» [140].

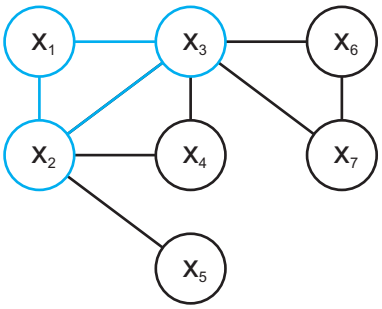


Рисунок 1.1: Пример графа взаимосвязей

Определение 1.6. Последовательность вершин, в которой каждая текущая вершина взаимосвязана со следующей, называется путь в графе.

Перейдём к определению цикла в графе [140].

Определение 1.7. Путь, для которого начальная и конечная вершины совпадают, называется **циклом**.

Рассмотрим определение связного графа [140].

Определение 1.8. Граф, для которого существует путь из каждой вершины в каждую, называется связным.

Очевидно, что исходя из размерности матрицы $A_{N\times M}$ и числа ненулевых элементов $\mathfrak{z}$ нельзя сделать однозначный вывод о связности соответствующего графа взаимосвязей. Но при этом можно сформулировать условие, при котором граф всегда будет несвязным. Сформулируем необходимый признак связности в графе взаимосвязей.

Теорема 1.1. Для того, чтобы граф взаимосвязей $G(X, E)$ был связным необходимо, чтобы для каждой строки i соответствующей матрицы $A_{N\times M}$ существовал хотя бы один связывающий столбец.

Доказательство. Пусть дан связный граф взаимосвязей $G(X, E)$ и матрица $A_{N\times M}$, для которой для каждой строки существует хотя бы один связывающий столбец. Для доказательства воспользуемся методом от противного. Предположим, связному графу $G(X, E)$ не обязательно соответствует матрица с заданными условиями. Значит, согласно определению 1.4, в соответствующей графу матрице $A'_{N\times M}$ существует строка, для которой нет ни одного связывающего столбца. Рассмотрим компоненту графа, соответствующую данной строке. Данная компонента не содержит рёбер, которые связывают её с другими частями

графа, что нарушает связность графа по определению 1.8. Значит предположение неверно и матрица $A'_{N\times M}$ не может являться матрицей, соответствующей связному графу. Таким образом, матрицей, соответствующей графу $G(X,E)$ является матрица $A_{N\times M}$. Теорема доказана.

Далее перейдём к теореме, которая является достаточным условием для того, чтобы граф взаимосвязей не был связным.

Теорема 1.2. *$n+m-1>\mathfrak{z}\Rightarrow G(X,E)$ не связный.*

Доказательство. Даны граф взаимосвязей и соответствующая ему матрица $A_{N\times M}$, содержащая $\mathfrak{z}$ ненулевых элементов. Каждый столбец должен содержать хотя бы один элемент, поскольку матрица не содержит пустых столбцов. Общее число столбцов — n, поэтому если каждый столбец содержит по одному элементу, число ненулевых элементов $\mathfrak{z}=n$, при этом каждый из этих столбцов не будет связывающим. Согласно теореме 1.1, каждая строка матрицы $A_{N\times M}$ должна содержать связывающий столбец 1.4. Число связывающих столбцов в матрице, как минимум, $m-1$, поскольку связывающий столбец соответствует хотя бы двум строкам в матрице, а всего матрица содержит $m-1$ пар строк. Таким образом, минимальное число ненулевых элементов в матрице $A_{N\times M}$, при котором граф может быть связным, равно $n+m-1$. Значит, для того, чтобы граф взаимосвязей не был связным, достаточно, чтобы число ненулевых элементов подчинялось неравенству $\mathfrak{z}<n+m-1$. Теорема доказана.

Перейдём к определению дерева [140].

Определение 1.9. Деревом называется связный граф без циклов.

Рассмотрим критерий существования дерева [141].

Теорема 1.3. *Граф является деревом тогда и только тогда, когда число его рёбер равно $n-1$.*

Введём понятие окрестности для столбца матрицы A в графовом представлении [81].

Определение 1.10. Множество вершин, то есть связанных ребром с вершиной x в графе взаимосвязей $G(X,E)$, обозначается $Nb(x)$ и называется окрестностью вершины x.

Замечание. Для практических задач понятие окрестности достаточно тонкий вопрос, поскольку по факту вершины с близкими окрестностями объединяют в большие окрестности относительно групп вершин. Под близкими окрестностями подразумевается, что пересечение соответствующих множеств значительно больше их разности. Укрупнение происходит в зависимости от размерности матрицы A, числа ненулевых элементов в ней, а также возможностей вычислительной системы. Подробнее этот аспект будет рассматриваться во второй главе при непосредственном выделении структуры разреженной матрицы.

Определим граф пересечения окрестностей [81]. Представим граф $G(X, E)$ в виде системы окрестностей $\Omega_1 = (S_1, U_1),\ \Omega_2 = (S_2, U_2), \ldots,\ \Omega_k = (S_k, U_k)$ вершин $x_{j_1}, \ldots, x_{j_k}$, где S_r и U_r — множества номеров столбцов и строк соответствующей матрицы относительно r–й окрестности, $r = 1, \ldots, k$.

Определение 1.11. *Графом пересечений окрестностей G_Ω называется граф, вершины которого ν_{r_i} — окрестности $\Omega_{r_i} = (S_{r_i}, U_{r_i})$, при этом вершины графа ν_{r_1} и ν_{r_2}, соединяются ребром $(r_1,\ r_2)$, если $S_{r_1} \bigcap S_{r_2} \neq \emptyset$.*

Пусть для системы окрестностей графа $G(X, E)$ выполняются следующие свойства:

— объединение множеств номеров строк для каждой окрестности соответствует множеству номеров строк всей матрицы

$$\bigcup_{r=1}^{k} U_r = M = \{1, \ldots, m\}; \tag{1.1}$$

— объединение множеств номеров столбцов для каждой окрестности соответствует множеству номеров столбцов всей матрицы

$$\bigcup_{r=1}^{k} S_r = N = \{1, \ldots, n\}; \tag{1.2}$$

— множества номеров столбцов, соответствующие любым двум окрестностям из заданной системы окрестностей не пересекаются

$$U_{r_1} \bigcap U_{r_2} = \emptyset, r_1 \neq r_2; \tag{1.3}$$

— множества номеров строк, соответствующие любым трём окрестностям из заданной системы окрестностей не пересекаются одновременно

$$S_{r_1} \bigcap S_{r_2} \bigcap S_{r_3} = \emptyset \tag{1.4}$$

для любых r_1, r_2, r_3. Перейдём к определению блочно–древовидной структуры матрицы A (БД–структуры) [81].

Определение 1.12. Граф G_Ω, для которого выполняются свойства 1.1–1.4, и при этом он является деревом, называется блочно–древовидной структурой (БД–структурой).

Матрицу, соответствующую БД–структуре будем называть блочно–древовидной. Выделенную вершину ν_r будем называть корнем T. БД–структура определяется соотношением «предок — потомок». При этом, если $(\nu_r, \nu_1), \ldots, (\nu_{p-1}, \nu_p)$ — путь от корня ν_r в вершину ν_p,то ν_{p-1} называют предком вершины ν_p, а ν_p — потомком вершины ν_{p-1}.

Определение 1.13. Граф G_Ω, для которого выполняются свойства 1.1–1.4, и при этом он является цепью, называется блочно–лестничной структурой (БЛ–структурой).

Матрицу, соответствующую БЛ–структуре будем называть блочно–лестничной.

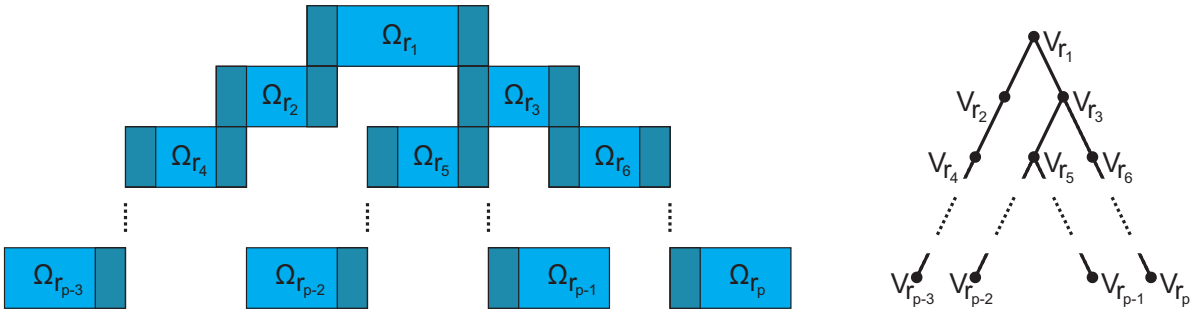


Рисунок 1.2: БД–структура

Определим компоненты БД–структуры. Введём определение блока.

Определение 1.14. Вершины ν_{r_i}, соответствующие каждой r–й окрестности, будем называть блоками БД–структуры, количество блоков обозначим $\mathtt{k}$.

Введём определение степени БД–структуры [81].

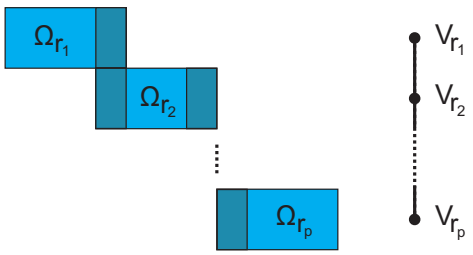


Рисунок 1.3: БЛ–структура

Определение 1.15. *Степенью БД–структуры будем называть* $\rho = \max\{\rho_1, \ldots, \rho_k\}$*, где* $\rho_1, \ldots, \rho_k$ *— степени каждого блока в БД–структуре.*

Введём определение сепаратора [81].

Определение 1.16. Для данного связного графа $G(X, E)$ и множества индексов вершин $S \subset X$, таких что подграф $G[X \setminus S]$ графа $G(X, E)$ несвязен, S называется **сепаратором**, если .

Введём метрику для БД–структуры. Определим расстояние между блоками.

Определение 1.17. Расстоянием между двумя вершинами БД–структуры будем называть минимальное число рёбер пути, соединяющего эти вершины.

Введём понятие слоя БД–структуры.

Определение 1.18. Множество вершин, которые находятся на одном расстоянии от корня, будем называть слоем БД–структуры.

Каждый блок содержит сепараторы, которые входят ещё в какой–нибудь блок, и не–сепараторы, то есть множества индексов вершин, принадлежащие только данному блоку. Обратим внимание, что граф G_Ω может быть несвязным. Тогда БД–структура распадается на независимые блоки. Для задач большой размерности структуры таких блоков имеет смысл рассматривать отдельно, так как каждый такой независимый блок может оказаться БД–структурой с достаточно большим числом вершин. Исходя из размерности матрицы и числа её ненулевых элементов можно сформулировать необходимое условие связности БД–структуры.

Теорема 1.4. Для того, чтобы БД–структура, соответствующая графу окрестностей G_Ω, была связной, необходимо, чтобы $n + m - 1 \leq \mathfrak{z}$.

Доказательство. Согласно теореме 1.2 граф взаимосвязей $G(X, E)$ несвязен при $n + m - 1 > \mathfrak{z}$. Значит существует как минимум одна вершина x_j, для которой нет соседних вершин. Значит данная вершина не входит ни в одну окрестность. По определению, вершины ν_{r_i} графа пересечения окрестностей G_Ω соответствуют каждой r–й окрестности, причем две вершины графа ν_{r_1} и ν_{r_2}, соединяются ребром $(r_1,\ r_2)$, если $S_{r_1} \bigcap S_{r_2} \neq \emptyset$. x_j может быть вершиной ν_{x_j}, но не может соединяться ребром с какой-либо другой вершиной графа G_Ω, поскольку $S_{x_j} \bigcap S_{\bar{x_j}} = \emptyset$. Существование хотя бы одной вершины, не связанной с другими вершинами графа G_Ω по определению 1.8 делает граф G_Ω несвязным. Значит и соответствующий ему БД–структура — несвязная. Значит, чтобы БД–была связна, необходимо, чтобы условие $n+m+1 > \mathfrak{z}$ нарушалось: $n+m-1 \leq \mathfrak{z}$. Теорема доказана.

Не зная структуры графа, нельзя предположить, является он деревом или нет, но можно определить соотношение между степенью графа и количеством блоков. Сформулируем теорему о связи компонент БД–структуры $\mathtt{k}$ и ρ.

Теорема 1.5. Степень БД–структуры всегда меньше числа её блоков.

Доказательство. Пусть G_Ω — данная БД–структура. По определению 1.12, она является деревом. Согласно критерию определения дерева 1.3, число рёбер БД–структуры должно быть меньше числа блоков на единицу. Наиболее разветвлённая БД–структура, состоящая из $\mathtt{k}$ блоков, имеет вид корня, с которым связаны все остальные $\mathtt{k} - 1$ вершин. Степень корня при этом является максимально возможной степенью относительно других блоков и равна $\mathtt{k} - 1$. По определению 1.15, это число является степенью предложенной БД–структуры. Менее разветвлённые БД–структуры будут иметь ещё меньшую степень. Значит, $\rho \leq \mathtt{k} - 1$. Теорема доказана.

Заметим, что число нулевых элементов в БД–структуре подчиняется некоторому соотношению с размерностью матрицы и характеристиками БД–структуры, такими как степень БД–структуры и число блоков, из которых БД–структура состоит. Это соотношение задано в явном виде согласно необходимому условию выделяемости БД–структуры О.А. Щербины [81].

Теорема 1.6. Если A — матрица $N \times M$ с $\mathfrak{z}_0$ нулевыми элементами, то для того, чтобы она имела БД–структуру с $\mathtt{k}$ блоками, необходимо, чтобы $n \geq 2\mathtt{k} - 1$,

$m \geq \mathtt{k}$ и

$$\mathfrak{z}_0 \geq (\mathtt{k}-2)(2m-n-2\mathtt{k}+\rho+2)-m(\rho-2)-3\mathtt{k}+2\rho+4, \mathtt{k} \geq 2$$

Замечание. Случай $\mathtt{k} = 2$ не представляет интерес для матриц большой размерности, так как декомпозиция всего на два блока не даёт существенный выигрыш по времени для решения соответствующей задачи. Будем рассматривать случай $\mathtt{k} > 2$.

Исследуем подробнее соотношение $\mathfrak{z} \geq (\mathtt{k}-2)(2m-n-2\mathtt{k}+\rho+2)-m(\rho-2)-3\mathtt{k}+2\rho+4$. Легко заметить, что для ненулевых элементов в матрице данное соотношение будет иметь вид

$$m+(m-\mathtt{k}+2)(n-2\mathtt{k}+\rho+1)+2(\mathtt{k}-1-\rho) \leq \mathfrak{z}$$

Преобразуем данное неравенство так, чтобы выразить в явном виде компоненты БД–структуры.

$$\rho \leq (-2\mathtt{k}^2+(n+2m+3)\mathtt{k}+\mathfrak{z}-2m-2n-mn)/(m-\mathtt{k})$$

Далее сформулируем модифицированное необходимое условие выделяемости БД–структуры, включающее видоизменённое неравенство.

Теорема 1.7. *Если A — матрица $N \times M$ с $\mathfrak{z}$ ненулевыми элементами, то для того, чтобы она имела БД–структуру с $\mathtt{k}$ блоками, необходимо, чтобы $n \geq 2\mathtt{k}-1$, $m \geq \mathtt{k}$ и*

$$\rho \leq (-2\mathtt{k}^2+(n+2m+3)\mathtt{k}+\mathfrak{z}-2m-2n-mn)/(m-\mathtt{k})$$

Доказательство.

$$m+(m-\mathtt{k}+2)(n-2\mathtt{k}+\rho+1)+2(\mathtt{k}-1-\rho) \leq \mathfrak{z}$$

$$m+mn-2m\mathtt{k}+m\rho+m-\mathtt{k}n+2\mathtt{k}^2-\mathtt{k}\rho-\mathtt{k}+2n-4\mathtt{k}+2\rho-2\rho+2\mathtt{k}+2-2-\mathfrak{z} \leq 0$$

$$m\rho-\mathtt{k}\rho \leq -2\mathtt{k}^2+(2m+n+3)\mathtt{k}-2m-2n-mn+\mathfrak{z}$$

$$\rho \leq 1/m - \mathtt{k}(-2\mathtt{k}^2 + (2m + n - 3)\mathtt{k} - 2m - 2n - mn + \mathfrak{z})$$

Теорема доказана.

1.2 Зависимость параметров в БД– и БЛ–структурах

Далее вводятся и доказываются теоремы, в которых устанавливается связь между числом блоков и степенью БД– и БЛ–структур в зависимости от количества ненулевых элементов в матрице и её размерности. Рассмотрим соотношение из теоремы 1.7 для каждой из типов матриц согласно классификации (1.1). Сформулируем лемму об определении аналитической формы области определения БД–структуры.

Лемма 1.1. Область определения БД–структуры разреженной матрицы задаётся следующими нелинейными неравенствами:

$$\begin{cases} \rho \leq (-2\mathtt{k}^2 + (n + 2m + 3)\mathtt{k} + \mathfrak{z} - 2m - 2n - mn)/(m - \mathtt{k}), \\ 2 \leq \rho \leq \mathtt{k} - 1, \\ 3 \leq \mathtt{k} < \min(m, 0.5(n + 1)), \end{cases}$$

где параметры подчиняются соотношениям $n + m - 1 \leq \mathfrak{z} < 0.5mn$, $m > 3$, $n > 3$.

Доказательство. Пусть дана разреженная матрица. По определению разреженности 1.3 $\mathfrak{z} < 0.5mn$. Исходя из того, что из данной матрицы можно извлечь БД–структуру, должно выполняться модифицированное необходимое условие 1.7 $\rho \leq (-2\mathtt{k}^2 + (n + 2m + 3)\mathtt{k} + \mathfrak{z} - 2m - 2n - mn)/(m - \mathtt{k})$ при $3 \leq \mathtt{k} < \min(m, 0.5(n + 1))$. По определению 1.12, БД–структура должна быть связной и не иметь циклов. Из необходимого условия связности 1.4 следует, что $n + m - 1 \leq \mathfrak{z}$. Согласно теореме 1.5, компоненты БД–структуры должны быть связаны следующим образом: $\rho \leq \mathtt{k} - 1$. Теорема доказана.

Далее приведём ряд теорем об аналитической форме области определения компонент БД–структуры. Вначале сформулируем теорему для случая очень широкой матрицы.

Теорема 1.8. Область определения компонент БД–структуры для очень широкой матрицы задаётся следующими нелинейными неравенствами:

$$\begin{cases} \rho \le (-2\mathtt{k}^2 + (n + 2m + 3)\mathtt{k} + \mathfrak{z} - 2m - 2n - mn)/(m - \mathtt{k}), \\ 2 \le \rho \le \mathtt{k} - 1, \\ 3 \le \mathtt{k} < m, \end{cases}$$

где параметры подчиняются соотношениям $n + m - 1 \le \mathfrak{z} < 0.5mn$, $3 < m < 0.5(n + 1)$.

Доказательство. Пусть дана разреженная матрица с очень широкой матрицей $A_{M\times N}$. Согласно классификации 1.1 это означает, что её параметры определяются соотношением $m < 0.5(n + 1)$. Значит $\mathtt{k} < m$. Остальные неравенства определяются согласно лемме 1.1. Теорема доказана.

Аналогичным образом доказываются теоремы 1.9, 1.10, 1.11, где формулируется аналитическая форма области определения компонент БД–структуры для случаев широкой, квадратной и узкой матрицы соответственно. Далее рас-

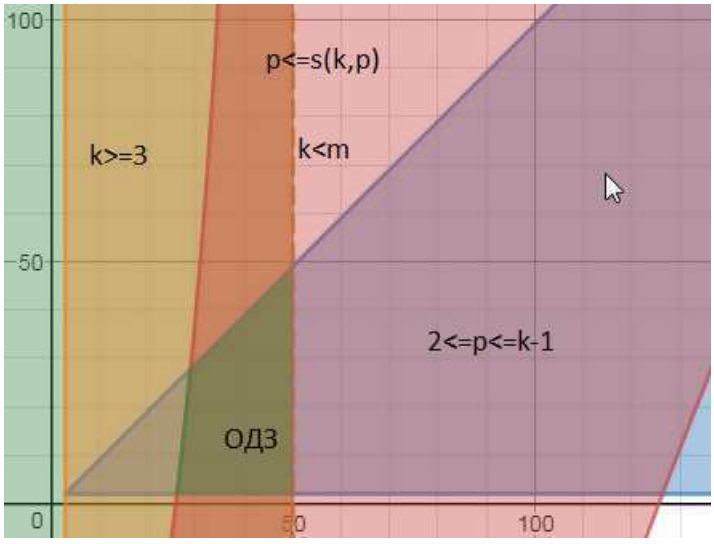


Рисунок 1.4: Область определения для очень широкой матрицы $n = 200, m = 50, \mathfrak{z} = 4000$

смотрим аналитическую форму области определения компонент БД–структуры для широкой матрицы, сформулированную в следующей теореме.

Теорема 1.9. Область определения компонент БД–структуры для широкой матрицы задаётся нелинейными неравенствами:

$$\begin{cases} \rho \le (-2\mathtt{k}^2 + (n + 2m + 3)\mathtt{k} + \mathfrak{z} - 2m - 2n - mn)/(m - \mathtt{k}), \\ 2 \le \rho \le \mathtt{k} - 1, \\ 3 \le \mathtt{k} < 0.5(n + 1), \end{cases}$$

где параметры подчиняются соотношениям $n+m-1 \le \mathfrak{z} < 0.5mn$, $0.5(n+1) \le m < n$, $m \ge 3$.

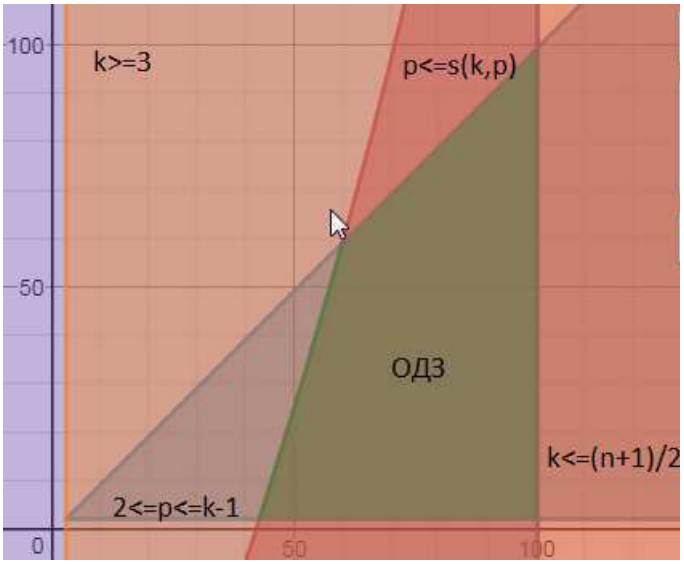


Рисунок 1.5: Область определения для широкой матрицы $n = 200, m = 150, \mathfrak{z} = 13000$

Затем сформулируем теорему для случая квадратной матрицы.

Теорема 1.10. Область определения компонент БД–структуры для квадратной матрицы задаётся нелинейными неравенствами:

$$\begin{cases} \rho \le (-2\mathtt{k}^2 + (3n + 3)\mathtt{k} + \mathfrak{z} - 4n - n^2)/(n - \mathtt{k}), \\ 2 \le \rho \le \mathtt{k} - 1, \\ 3 \le \mathtt{k} < 0.5(n + 1), \end{cases}$$

где параметры подчиняются соотношению $2n - 1 < \mathfrak{z} < 0.5n^2$.

Наконец сформулируем теорему для случая узкой матрицы.

Теорема 1.11.

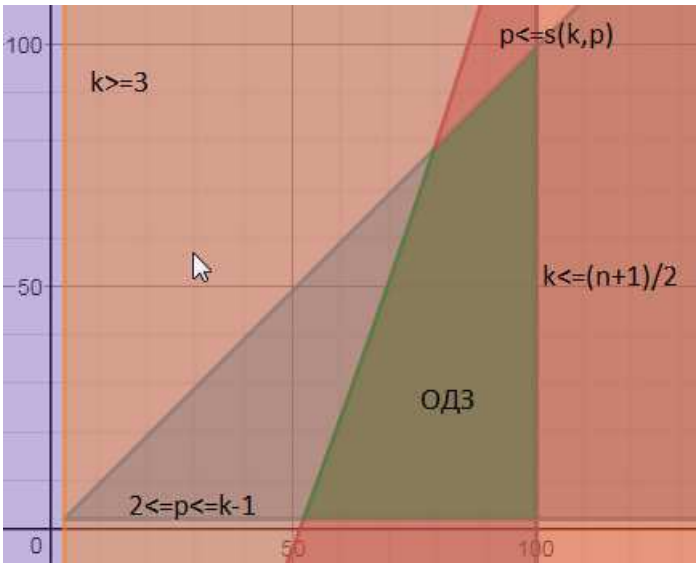


Рисунок 1.6: Область определения для квадратной матрицы $n = m = 200, \mathfrak{z} = 15000$

Область определения компонент БД–структуры для узкой матрицы задаётся следующими соотношениями:

$$\begin{cases} \rho \le (-2\mathtt{k}^2 + (n + 2m + 3)\mathtt{k} + \mathfrak{z} - 2m - 2n - mn)/(m - \mathtt{k}), \\ 2 \le \rho \le \mathtt{k} - 1, \\ 3 \le \mathtt{k} < 0.5(n + 1), \end{cases}$$

где параметры подчиняются соотношениям $n + m - 1 \le \mathfrak{z} < 0.5mn$, $3 \le n < m$.

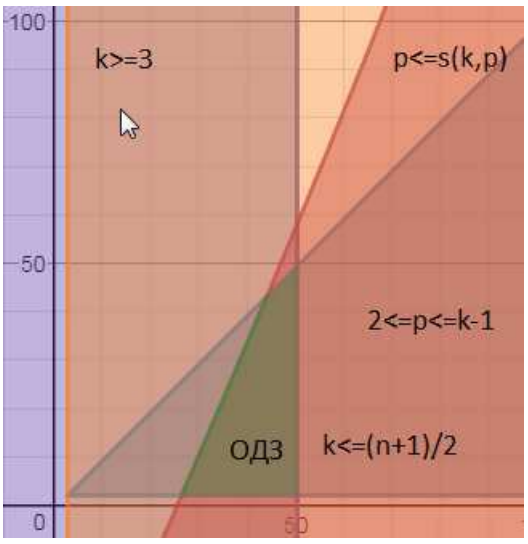


Рисунок 1.7: Область определения для узкой матрицы $n = 100, m = 150, \mathfrak{z} = 9000$

Полученные области позволяют определить возможные компоненты БД–структуры для заданного графа, если мы знаем, что она существует. То есть, если БД–структуру можно выделить, данные теоремы помогают сделать оценку для числа блоков $\mathtt{k}$ и степени БД–структуры ρ относительно размерности матрицы и числа её значимых элементов $\mathfrak{z}$. Рассмотрим частный случай БД–структуры — БЛ–структуру. Определим, в каких границах находится число блоков БЛ–структуры. Для этого сформулируем утверждение о числе блоков в БЛ–структуре в общем случае.

Лемма 1.2. Число блоков БЛ–структуры ограничивается

1. В случае $m \geq 0.5(n+1)$
   - $3 \leq \mathtt{k} < 0.5(n+1)$ при $m > 2$, $n > 5$, $\mathfrak{z} \geq 0.5(2mn - 6m - 3n + 14)$
   - $0.25(n + 2m + 3 - \sqrt{(n-2m+3)^2 + 8\mathfrak{z} - 12n + 16}) \leq \mathtt{k} < 0.5(n+1)$ при $m > 2$, $n \geq 2m+1$, $0.125(12n - 16 - (n-2m+3)^2) < \mathfrak{z} \leq 0.5(2mn - 6m - 3n + 14)$
2. В случае $m < 0.5(n+1)$
   - $3 \leq \mathtt{k} < m$ при $m > 3$, $n > 2m-1$, $\mathfrak{z} > 0.5(2mn - 6m - 3n + 14)$
   - $0.25(n + 2m + 3 - \sqrt{(n-2m+3)^2 + 8\mathfrak{z} - 12n + 16}) \leq \mathtt{k} < m$ при $m > 3$, $n > 2m-1$, $0.125(12m+6n-25-(n-2m)^2) \leq \mathfrak{z} \leq 0.5(2mn - 6m - 3n + 14)$
   - $0.25(n + 2m + 3 - \sqrt{(n-2m+3)^2 + 8\mathfrak{z} - 12n + 16}) \leq \mathtt{k} < m$ при $m > 3$, $5 < n \leq 2m-1$, $0.5(4m+n-6) < \mathfrak{z} \leq 0.5(2mn-6m-3n+14)$

При этом значимые элементы матрицы $\mathfrak{z}$ и размерность матрицы n, m соотносятся следующим образом: $n + m - 1 \leq \mathfrak{z} < 0.5mn$, $m > 3$, $n > 3$.

Доказательство. Согласно лемме 1.1, область определения БД–структуры в общем виде задаётся следующими нелинейными неравенствами:

$$\begin{cases} \rho \leq (-2\mathtt{k}^2 + (n+2m+3)\mathtt{k} + \mathfrak{z} - 2m - 2n - mn)/(m - \mathtt{k}), \\ 2 \leq \rho \leq \mathtt{k} - 1, \\ 3 \leq \mathtt{k} < \min(m, 0.5(n+1)), \end{cases}$$

где параметры подчиняются соотношениям $n + m - 1 \leq \mathfrak{z} < 0.5mn$, $m > 3$, $n > 3$. Для БЛ–структуры $\rho = 2$. Значит соотношение примет вид:

$$\begin{cases} 2 \leq (-2\mathtt{k}^2 + (n + 2m + 3)\mathtt{k} + \mathfrak{z} - 2m - 2n - mn)/(m - \mathtt{k}), \\ 3 \leq \mathtt{k} < \min(m, 0.5(n + 1)), \end{cases}$$

где параметры подчиняются соотношениям $n + m - 1 \leq \mathfrak{z} < 0.5mn$, $m > 3$, $n > 3$. Исследуем первое неравенство системы.

$2 \leq (-2\mathtt{k}^2 + (n + 2m + 3)\mathtt{k} + \mathfrak{z} - 2m - 2n - mn)/(m - \mathtt{k})$,

$-2\mathtt{k}^2 + (n + 2m + 5)\mathtt{k} + \mathfrak{z} - 4m - 2n - mn \geq 0$

Решением данного неравенства будет:

$\mathtt{k} \in [0.25(n + 2m + 3 - \sqrt{(n - 2m + 3)^2 + 8\mathfrak{z} - 12n + 16}); 0.25(n + 2m + 3 + \sqrt{(n - 2m + 3)^2 + 8\mathfrak{z} - 12n + 16})]$

Таким образом, получим систему:

$$\begin{cases} 0.25(n + 2m + 3 - \sqrt{(n - 2m + 3)^2 + 8\mathfrak{z} - 12n + 16}) \leq \mathtt{k} \leq 0.25(n + 2m + 3 + \sqrt{(n \\ 3 \leq \mathtt{k} < \min(m, 0.5(n + 1)). \end{cases}$$

Верхняя граница всегда $\mathtt{k} < \min(m, 0.5(n + 1)) < 0.25(n + 2m + 3 + \sqrt{(n - 2m + 3)^2 + 8\mathfrak{z} - 12n + 16})$. Исследуем нижнюю границу.

$$\begin{cases} 0.25(n + 2m + 3 - \sqrt{(n - 2m + 3)^2 + 8\mathfrak{z} - 12n + 16}) \leq \mathtt{k}, \\ 3 \leq \mathtt{k} < \min(m, 0.5(n + 1)), \\ m > 3. \end{cases}$$

Решениями данной системы будут:

1. В случае $m \geq 0.5(n + 1)$
   - $3 \leq \mathtt{k} < 0.5(n + 1)$ при $m > 2$, $n > 5$, $\mathfrak{z} \geq 0.5(2mn - 6m - 3n + 14)$
   - $0.25(n + 2m + 3 - \sqrt{(n - 2m + 3)^2 + 8\mathfrak{z} - 12n + 16}) \leq \mathtt{k} < 0.5(n + 1)$ при $m > 2$, $n \geq 2m + 1$, $0.125(12n - 16 - (n - 2m + 3)^2) < \mathfrak{z} \leq 0.5(2mn - 6m - 3n + 14)$
2. В случае $m < 0.5(n + 1)$
   - $3 \leq \mathtt{k} < m$ при $m > 3$, $n > 2m - 1$, $\mathfrak{z} > 0.5(2mn - 6m - 3n + 14)$

— $0.25(n+2m+3-\sqrt{(n-2m+3)^2+8\mathfrak{z}-12n+16}) \le \mathtt{k} < m$ при $m>3$, $n>2m-1$, $0.125(12m+6n-25-(n-2m)^2) \le \mathfrak{z} \le 0.5(2mn-6m-3n+14)$

— $0.25(n+2m+3-\sqrt{(n-2m+3)^2+8\mathfrak{z}-12n+16}) \le \mathtt{k} < m$ при $m>3$, $5<n\le 2m-1$, $0.5(4m+n-6) < \mathfrak{z} \le 0.5(2mn-6m-3n+14)$

Теорема доказана.

Далее сформулируем теорему, основанную на лемме 1.2, о числе блоков в БЛ–структуре для очень широкой матрицы.

Теорема 1.12. Число блоков БЛ–структуры для очень широкой матрицы ограничивается $0.25(n+2m+3-\sqrt{(n-2m+3)^2+8\mathfrak{z}-12n+16}) \le \mathtt{k} < m$, где значимые элементы матрицы $\mathfrak{z}$ и размерность матрицы n, m соотносятся следующим образом:

— $m \ge 4$, $n > 2m+2$, $0.5(3n-4) < \mathfrak{z} < 0.5mn$

— $m \ge 4$, $2m \le n \le 2m+2$, $m+n-1 \le \mathfrak{z} < 0.5mn$

Доказательство. Согласно классификации матриц 1.1, для очень широких матриц имеет место соотношение $3 < m < \frac{n+1}{2}$. Значит $3 \le \mathtt{k} < m$ и для числа блоков согласно лемме 1.2 имеют место следующие соотношения:

— $3 \le \mathtt{k} < m$ при $m>3$, $n>2m-1$, $\mathfrak{z} > \frac{1}{2}(2mn-6m-3n+14)$

— $\frac{1}{4}(n+2m+3-\sqrt{(n-2m+3)^2+8\mathfrak{z}-12n+16}) \le \mathtt{k} < m$ при $m>3$, $n>2m-1$, $\frac{1}{2}(3n-4) < \mathfrak{z} \le \frac{1}{2}(2mn-6m-3n+14)$

При этом значимые элементы матрицы $\mathfrak{z}$ и размерность матрицы n, m соотносятся следующим образом: $n+m-1 \le \mathfrak{z} < \frac{mn}{2}$ согласно лемме 1.1, $3 < m < \frac{n+1}{2}$, $n > 3$ согласно классификации матриц 1.1. Добавим ограничения по $\mathfrak{z}$:

—

$$\begin{cases} \mathfrak{z} > \frac{1}{2}(2mn-6m-3n+14), \\ n+m-1 \le \mathfrak{z} < \frac{mn}{2}, \\ 3 < m < \frac{n+1}{2}, \\ n > 3. \end{cases}$$

Решение данной системы $m=4$, $n=9, z=18$

—
$$\begin{cases} \frac{1}{2}(3n-4) < \mathfrak{z} \leq \frac{1}{2}(2mn-6m-3n+14), \\ n+m-1 \leq \mathfrak{z} < \frac{mn}{2}, \\ 3 < m < \frac{n+1}{2}, \\ n > 3. \end{cases}$$

Решения данной системы:

— $m \geq 4$, $n > 2m+2$, $\frac{1}{2}(3n-4) < z < mn/2$

— $m \geq 4$, $2m \leq n \leq 2m+2$, $m+n-1 \leq z < mn/2$

Таким образом, решения примут вид:

— $\mathtt{k} = 3$ при $m = 4$, $n = 9$, $\mathfrak{z} = 18$

— $\frac{1}{4}(n+2m+3-\sqrt{(n-2m+3)^2+8\mathfrak{z}-12n+16}) \leq \mathtt{k} < m$ при $m \geq 4$, $n > 2m+2$, $\frac{1}{2}(3n-4) < z < mn/2$

— $\frac{1}{4}(n+2m+3-\sqrt{(n-2m+3)^2+8\mathfrak{z}-12n+16}) \leq \mathtt{k} < m$ при $m \geq 4$, $2m \leq n \leq 2m+2$, $m+n-1 \leq z < mn/2$

Заметим, что первое решение поглощается, значит окончательное множество решений примет вид:

— $\frac{1}{4}(n+2m+3-\sqrt{(n-2m+3)^2+8\mathfrak{z}-12n+16}) \leq \mathtt{k} < m$ при $m \geq 4$, $n > 2m+2$, $\frac{1}{2}(3n-4) < z < \frac{mn}{2}$

— $\frac{1}{4}(n+2m+3-\sqrt{(n-2m+3)^2+8\mathfrak{z}-12n+16}) \leq \mathtt{k} < m$ при $m \geq 4$, $2m \leq n \leq 2m+2$, $m+n-1 \leq z < \frac{mn}{2}$

Аналогичным образом доказывается следующая лемма 1.3 о числе блоков в БЛ–структуре для классов матриц, не относящихся к очень широким.

Лемма 1.3. Число блоков БЛ–структуры для матрицы, не относящейся к классу очень широких матриц, ограничивается:

— $3 \leq \mathtt{k} < 0.5(n+1)$ при:

1) $m = 4$, $n \in \{6; 7\}$, $2.5(n-2) \le \mathfrak{z} \le 2n-1$;

2) $m \ge 5$, $6 \le n < 2(3m-7)/(m-3)$, $0.5(2mn-6m-3n+14) \le \mathfrak{z} \le 0.5mn$.

— $0.25(n+2m+3-\sqrt{(n-2m+3)^2+8\mathfrak{z}-12n+16}) \le \mathtt{k} < 0.5(n+1)$ при:

1) $4 \le m \le 5$, $n = 11-m$, $m+6 \le \mathfrak{z} \le m+8$

2) $m = 5$, $7 \le n \le 9$, $n+4 \le \mathfrak{z} \le 3n-m$

3) $m = 7$, $n = 7$, $15 \le \mathfrak{z} \le 24$

4) $m \ge 6$, $2m-3 \le n \le 2m-1$, $m+n-1 \le \mathfrak{z} < 0.5mn$

5) $m \ge 6$, $2(3m-7)/(m-3) \le n \le 2m-4$, $0.5(4m+n-6) < \mathfrak{z} < 0.5mn$

6) $m \ge 6$, $6 \le n < 2(3m-7)/(m-3)$, $0.5(4m+n-6) < \mathfrak{z} \le 0.5(2mn-6m-3n+14)$

На основе леммы 1.3 сформулируем теорему о числе блоков в БЛ–структуре для широкой матрицы.

Теорема 1.13. Число блоков БЛ–структуры для широкой матрицы ограничивается:

1. $3 \le \mathtt{k} < 0.5(n+1)$, если параметры матрицы имеют вид:

   1) $m = 4$, $6 \le n \le 7$, $2.5(n-2) \le \mathfrak{z} \le 2n-1$

   2) $m = 5$, $6 \le n \le 7$, $0.5(7n-16) \le \mathfrak{z} \le 2.5n$

   3) $m = 6$, $n = 7$, $\mathfrak{z} = 21$

2. $0.25(n+2m+3-\sqrt{(n-2m+3)^2+8\mathfrak{z}-12n+16}) \le \mathtt{k} < 0.5(n+1)$, если параметры матрицы имеют вид:

   1) $4 \le m \le 5$, $n = 11-m$, $m+6 \le \mathfrak{z} \le m+8$

   2) $m = 5$, $7 \le n \le 9$, $n+4 \le \mathfrak{z} \le 3n-5$

   3) $m = 6$, $7 \le n \le 8$, $n+6 \le \mathfrak{z} \le 3n-1$

   4) $m \ge 6$, $2m-3 \le n \le 2m-1$, $m+n-1 \le \mathfrak{z} < 0.5mn$

   5) $m \ge 7$, $m < n \le 2m-4$, $0.5(4m+n-6) < \mathfrak{z} < 0.5mn$

Доказательство. Согласно классификации матриц 1.1, для широких матриц имеют место соотношения $0.5(n+1) \leq m < n$, $m \geq 3$. Данный класс матриц не относится к классу очень широких матриц, значит, согласно лемме 1.3 значение числа блоков принимает значения:

— $3 \leq \mathtt{k} < 0.5(n+1)$ при:

1) $m = 4$, $n \in \{6; 7\}$, $2.5(n-2) \leq \mathfrak{z} \leq 2n-1$;
2) $m \geq 5$, $6 \leq n < 2(3m-7)/(m-3)$, $0.5(2mn - 6m - 3n + 14) \leq \mathfrak{z} \leq 0.5mn$.

— $0.25(n + 2m + 3 - \sqrt{(n-2m+3)^2 + 8\mathfrak{z} - 12n + 16}) \leq \mathtt{k} < m$ при:

1) $4 \leq m \leq 5$, $n = 11 - m$, $m + 6 \leq \mathfrak{z} \leq m + 8$
2) $m = 5$, $7 \leq n \leq 9$, $n + 4 \leq \mathfrak{z} \leq 3n - m$
3) $m = 7$, $n = 7$, $15 \leq \mathfrak{z} \leq 24$
4) $m \geq 6$, $2m - 3 \leq n \leq 2m - 1$, $m + n - 1 \leq \mathfrak{z} < 0.5mn$
5) $m \geq 6$, $2(3m-7)/(m-3) \leq n \leq 2m-4$, $0.5(4m+n-6) < \mathfrak{z} < 0.5mn$
6) $m \geq 6$, $6 \leq n < 2(3m-7)/(m-3)$, $0.5(4m+n-6) < \mathfrak{z} \leq 0.5(2mn - 6m - 3n + 14)$

Решим системы неравенств для каждого из этих случаев. Первый случай $3 \leq \mathtt{k} < 0.5(n+1)$, параметры примут вид

1)
$$\begin{cases} 2.5(n-2) \leq \mathfrak{z} \leq 2n-1, \\ 0.5(n+1) \leq m < n, \\ 6 \leq n \leq 7, \\ m = 4. \end{cases}$$

Решением данной системы будет: $m = 4$, $n \in \{6; 7\}$, $2.5(n-2) \leq \mathfrak{z} \leq 2n-1$

2)

$$\begin{cases} 0.5(2mn - 6m - 3n + 14) \le \mathfrak{z} \le 0.5mn, \\ 0.5(n+1) \le m < n, \\ 6 \le n < 2(3m-7)/(m-3), \\ m \ge 5. \end{cases}$$

Решениями данной системы будут:

— $m = 5$, $6 \le n \le 7$, $0.5(7n - 16) \le \mathfrak{z} \le 2.5n$

— $m = 6$, $n = 7$, $\mathfrak{z} = 21$

Второй случай $0.25(n + 2m + 3 - \sqrt{(n - 2m + 3)^2 + 8\mathfrak{z} - 12n + 16}) \le \mathtt{k} < m$, параметры примут вид

1)

$$\begin{cases} m + 6 \le \mathfrak{z} \le m + 8, \\ 0.5(n+1) \le m < n, \\ n = 11 - m, \\ 4 \le m \le 5. \end{cases}$$

Решение данной системы: $4 \le m \le 5$, $n = 11 - m$, $m + 6 \le \mathfrak{z} \le m + 8$

2)

$$\begin{cases} n + 4 \le \mathfrak{z} \le 3n - m, \\ 0.5(n+1) \le m < n, \\ 7 \le n \le 9, \\ m = 5. \end{cases}$$

Решение данной системы: $m = 5$, $7 \le n \le 9$, $n + 4 \le \mathfrak{z} \le 3n - m$

3)

$$\begin{cases} 15 \le \mathfrak{z} \le 24, \\ 0.5(n+1) \le m < n, \\ n = 7, \\ m = 7. \end{cases}$$

Данная система не имеет решений

4)

$$\begin{cases} m+n-1 \le \mathfrak{z} < 0.5mn, \\ 0.5(n+1) \le m < n, \\ 2m-3 \le n \le 2m-1, \\ m \ge 6. \end{cases}$$

Решение данной системы: $m \ge 6$, $2m-3 \le n \le 2m-1$, $m+n-1 \le \mathfrak{z} < 0.5mn$

5)

$$\begin{cases} 0.5(4m+n-6) < \mathfrak{z} < 0.5mn, \\ 0.5(n+1) \le m < n, \\ 2(3m-7)/(m-3) \le n \le 2m-4, \\ m \ge 6. \end{cases}$$

Решения данной системы:

— $m = 6$, $n = 8$, $0.5n + 9 < \mathfrak{z} < 3n$

— $m \ge 7$, $m < n \le 2m-4$, $0.5(4m+n-6) < \mathfrak{z} < 0.5mn$

6)

$$\begin{cases} 0.5(4m+n-6) < \mathfrak{z} \le 0.5(2mn-6m-3n+14), \\ 0.5(n+1) \le m < n, \\ 6 \le n < 2(3m-7)/(m-3), \\ m \ge 6. \end{cases}$$

$m = 6$, $n = 7$, $13 \le \mathfrak{z} \le 20$

Таким образом, число блоков в БЛ–структуре для широкой матрицы ограничивается:

1. $3 \le \mathtt{k} < 0.5(n+1)$, если параметры матрицы имеют вид:

   1) $m = 4$, $6 \le n \le 7$, $2.5(n-2) \le \mathfrak{z} \le 2n-1$

   2) $m = 5$, $6 \le n \le 7$, $0.5(7n-16) \le \mathfrak{z} \le 2.5n$

   3) $m = 6$, $n = 7$, $\mathfrak{z} = 21$

2. $0.25(n+2m+3-\sqrt{(n-2m+3)^2+8\mathfrak{z}-12n+16}) \le \mathtt{k} < m$, если параметры матрицы имеют вид:

   1) $4 \le m \le 5$, $n = 11-m$, $m+6 \le \mathfrak{z} \le m+8$
   2) $m = 5$, $7 \le n \le 9$, $n+4 \le \mathfrak{z} \le 3n-5$
   3) $m = 6$, $7 \le n \le 8$, $n+6 \le \mathfrak{z} \le 3n-1$
   4) $m \ge 6$, $2m-3 \le n \le 2m-1$, $m+n-1 \le \mathfrak{z} < 0.5mn$
   5) $m \ge 7$, $m < n \le 2m-4$, $0.5(4m+n-6) < \mathfrak{z} < 0.5mn$

Теорема доказана.

Аналогично доказываются теоремы 1.14, 1.15 о числе блоков в БЛ–структуре для квадратной и узкой матриц. Вначале рассмотрим теорему для случая квадратной матрицы.

Теорема 1.14. Число блоков БЛ–структуры для квадратной матрицы ограничивается $0.25(n+2m+3-\sqrt{(n-2m+3)^2+8\mathfrak{z}-12n+16}) \le \mathtt{k} < 0.5(n+1)$, если параметры матрицы имеют вид:

1. $3 \le \mathtt{k} < 0.5(n+1)$, если параметры матрицы имеют вид: $m = n = 6$, $16 \le \mathfrak{z} \le 18$.
2. $0.25(n+2m+3-\sqrt{(n-2m+3)^2+8\mathfrak{z}-12n+16}) \le \mathtt{k} < 0.5(n+1)$, если параметры матрицы имеют вид:

   1) $m = n = 6$, $13 \le \mathfrak{z} \le 16$
   2) $m = n \ge 7$, $0.5(5n-6) < \mathfrak{z} < 0.5n^2$

Далее сформулируем теорему в случае узкой матрицы.

Теорема 1.15. Число блоков БЛ–структуры для узкой матрицы ограничивается:

1. $3 \le \mathtt{k} < 0.5(n+1)$, если параметры матрицы имеют вид:

   1) $m = 7$, $n = 6$, $19 \le \mathfrak{z} \le 20$
   2) $m \ge 8$, $6 \le n < 2(3m-7)/(m-3)$, $0.5(2mn-6m-3n+14) \le \mathfrak{z} < 0.5mn$

2. $0.25(n+2m+3-\sqrt{(n-2m+3)^2+8\mathfrak{z}-12n+16}) \le \mathtt{k} < 0.5(n+1)$, если параметры матрицы имеют вид:

   1) $m=7$, $n=6$, $15 \le \mathfrak{z} \le 19$

   2) $m \ge 8$, $6 \le n < 2(3m-7)/(m-3)$, $0.5(4m+n-6) < \mathfrak{z} \le 0.5(2mn-6m-3n+14)$

Сформулируем лемму для определения границ аналитической области для матриц с БД–структурой.

Лемма 1.4. Аналитическая область для БД–структуры относительно числа блоков $\mathtt{k}$ определяется следующими соотношениями. Верхняя граница $\mathtt{k} < \min(m, 0.5(n+1))$. Нижняя граница выводится из соотношений:

Если $(-2\mathtt{k}^2+(n+2m+3)\mathtt{k}+\mathfrak{z}-2m-2n-mn)/(m-\mathtt{k}) = \mathtt{k}-1$

1) $3 \le 0.5(n+m+2-\sqrt{4(\mathfrak{z}-n+1)+(n-m)^2}) < \min(m, 0.5(n+1))$

2) $0.5(n+m+2-\sqrt{4(\mathfrak{z}-n+1)+(n-m)^2}) \ge \min(m, 0.5(n+1))$

Если $(-2\mathtt{k}^2+(n+2m+3)\mathtt{k}+\mathfrak{z}-2m-2n-mn)/(m-\mathtt{k}) = 2$

1) $3 \le 0.25(n+2m+5-\sqrt{4(2\mathfrak{z}-3n+4)+(n-2m+3)^2})$

2) $0.25(n+2m+5-\sqrt{4(2\mathfrak{z}-3n+4)+(n-2m+3)^2}) < 3$

При этом параметры матрицы подчиняются следующим соотношениям:

— $m \ge 5$, $n \ge 4$, $n+m-1 \le \mathfrak{z} < 0.5mn$

— $m=4$, $n \ge 4$, $n+3 \le \mathfrak{z} < 2n$

Доказательство. Согласно лемме 1.1, аналитическая форма области определения БД–структуры для матрицы общего вида выглядит следующим образом:

$$\begin{cases} \rho \le (-2\mathtt{k}^2+(n+2m+3)\mathtt{k}+\mathfrak{z}-2m-2n-mn)/(m-\mathtt{k}), \\ 2 \le \rho \le \mathtt{k}-1, \\ 3 \le \mathtt{k} < \min(m, 0.5(n+1)), \end{cases}$$

где параметры подчиняются соотношениям $n+m-1 \le \mathfrak{z} < 0.5mn$, $m > 3$, $n > 3$. Найдём точки пересечения кривых, определяющих данную область:

— Найдём пересечение $(-2\mathtt{k}^2 + (n + 2m + 3)\mathtt{k} + \mathfrak{z} - 2m - 2n - mn)/(m - \mathtt{k})$ и $\mathtt{k} - 1$

$(-2\mathtt{k}^2 + (n + 2m + 3)\mathtt{k} + \mathfrak{z} - 2m - 2n - mn)/(m - \mathtt{k}) = \mathtt{k} - 1$

$-2\mathtt{k}^2 + (n + 2m + 3)\mathtt{k} + \mathfrak{z} - 2m - 2n - mn = (\mathtt{k} - 1)(m - \mathtt{k})$

$-\mathtt{k}^2 + (n + m + 2)\mathtt{k} + \mathfrak{z} - m - 2n - mn = 0$

Решениями данного уравнения будут: $0.5(n + m + 2 - \sqrt{4(\mathfrak{z} - n + 1) + (n - m)^2})$ и $0.5(n + m + 2 + \sqrt{4(\mathfrak{z} - n + 1) + (n - m)^2})$.

Рассмотрим фрагмент $\mathtt{k} \in [0.5(n + m + 2 - \sqrt{4(\mathfrak{z} - n + 1) + (n - m)^2}); 0.5(n + m + 2 + \sqrt{4(\mathfrak{z} - n + 1) + (n - m)^2})]$ относительно промежутка области определения $\mathtt{k} \in [3; \min(m, 0.5(n+1)))$

Поскольку $0.5(n + m + 2 + \sqrt{4(\mathfrak{z} - n + 1) + (n - m)^2}) > \min m, 0.5(n + 1)$ $\Rightarrow$ данное значение выходит из области допустимых значений. Значит верхняя граница $\mathtt{k} < \min m, 0.5(n + 1)$.

Исследуем нижнюю границу для $\mathtt{k}$. Для второго значения существует два варианта:

1) $3 \le 0.5(n + m + 2 - \sqrt{4(\mathfrak{z} - n + 1) + (n - m)^2}) < \min(m, 0.5(n + 1))$
2) $0.5(n + m + 2 - \sqrt{4(\mathfrak{z} - n + 1) + (n - m)^2}) \ge \min(m, 0.5(n + 1))$

Подкоренное выражение $4(\mathfrak{z} - n + 1) + (n - m)^2 \ge 0$ выполняется при:

— $m \ge 5$, $n \ge 4$, $n + m - 1 \le \mathfrak{z} < 0.5mn$

— $m = 4$, $n \ge 4$, $n + 3 \le \mathfrak{z} < 2n$

— Найдём пересечение $(-2\mathtt{k}^2 + (n + 2m + 3)\mathtt{k} + \mathfrak{z} - 2m - 2n - mn)/(m - \mathtt{k})$ и 2.

$(-2\mathtt{k}^2 + (n + 2m + 3)\mathtt{k} + \mathfrak{z} - 2m - 2n - mn)/(m - \mathtt{k}) = 2.$

$-2\mathtt{k}^2 + (n + 2m + 3)\mathtt{k} + \mathfrak{z} - 2m - 2n - mn = 2m - 2\mathtt{k}$

$-2\mathtt{k}^2 + (n + 2m + 5)\mathtt{k} + \mathfrak{z} - 4m - 2n - mn = 0$

Решениями данного уравнения будут:$0.25(n + 2m + 5 + \sqrt{4(2\mathfrak{z} - 3n + 4) + (n - 2m + 3)^2})$ и $0.25(n + 2m + 5 - \sqrt{4(2\mathfrak{z} - 3n + 4) + (n - 2m + 3)^2})$.

Рассмотрим фрагмент $\mathtt{k} \in [0.25(n + 2m + 5 - \sqrt{4(2\mathfrak{z} - 3n + 4) + (n - 2m + 3)^2}); 0.25(n + 2m + 5 + \sqrt{4(2\mathfrak{z} - 3n + 4) + (n - 2m + 3)^2})]$ относительно промежутка области определения $\mathtt{k} \in [3; \min(m, 0.5(n + 1)))$

Поскольку $0.25(n + 2m + 5 + \sqrt{4(2\mathfrak{z} - 3n + 4) + (n - 2m + 3)^2}) > \min m, 0.5(n + 1) \Rightarrow$ данное значение выходит из области допустимых значений. Значит верхняя граница $\mathtt{k} < \min m, 0.5(n + 1)$

Исследуем нижнюю границу для $\mathtt{k}$. Для второго значения существует два варианта:

1) $3 \leq 0.25(n + 2m + 5 - \sqrt{4(2\mathfrak{z} - 3n + 4) + (n - 2m + 3)^2})$
2) $0.25(n + 2m + 5 - \sqrt{4(2\mathfrak{z} - 3n + 4) + (n - 2m + 3)^2}) < 3$

Подкоренное выражение $4(2\mathfrak{z} - 3n + 4) + (n - 2m + 3)^2 \geq 0$ выполняется при:

— $m \geq 5$, $n \geq 4$, $n + m - 1 \leq \mathfrak{z} < 0.5mn$

— $m = 4$, $n \geq 4$, $n + 3 \leq \mathfrak{z} < 2n$

Значит пересечения кривых определяются следующим образом. Верхняя граница $\mathtt{k} < \min m, 0.5(n + 1)$. Нижняя граница выводится из соотношений:

Если $(-2\mathtt{k}^2 + (n + 2m + 3)\mathtt{k} + \mathfrak{z} - 2m - 2n - mn)/(m - \mathtt{k}) = \mathtt{k} - 1$

1) $3 \leq 0.5(n + m + 2 - \sqrt{4(\mathfrak{z} - n + 1) + (n - m)^2}) < \min(m, 0.5(n + 1))$
2) $0.5(n + m + 2 - \sqrt{4(\mathfrak{z} - n + 1) + (n - m)^2}) \geq \min(m, 0.5(n + 1))$

Если $(-2\mathtt{k}^2 + (n + 2m + 3)\mathtt{k} + \mathfrak{z} - 2m - 2n - mn)/(m - \mathtt{k}) = 2$

1) $3 \leq 0.25(n + 2m + 5 - \sqrt{4(2\mathfrak{z} - 3n + 4) + (n - 2m + 3)^2})$
2) $0.25(n + 2m + 5 - \sqrt{4(2\mathfrak{z} - 3n + 4) + (n - 2m + 3)^2}) < 3$

При параметрах матрицы

— $m \geq 5$, $n \geq 4$, $n + m - 1 \leq \mathfrak{z} < 0.5mn$

— $m = 4$, $n \geq 4$, $n + 3 \leq \mathfrak{z} < 2n$

Теорема доказана.

Перейдём к более общему виду структуры матрицы — БД–структуре. Сформулируем необходимое условие выделяемости БД–структуры.

Лемма 1.5. Для того, чтобы в матрице общего вида можно было выделить БД–структуру, её параметры должны удовлетворять соотношениям $m \geq 4$, $n \geq 4$, $n+m-1 \leq \mathfrak{z} < 0.5mn$. При этом число блоков БД–структуры определяется следующим образом:

1) $3 \leq 0.25(n+2m+5-\sqrt{4(2\mathfrak{z}-3n+4)+(n-2m+3)^2}) \leq min(m, 0.5(n+1))$

2) $0.25(n+2m+5-\sqrt{4(2\mathfrak{z}-3n+4)+(n-2m+3)^2}) \leq 3 \leq min(m, 0.5(n+1))$

Доказательство. Согласно лемме 1.4, аналитическая область для БД–структуры относительно числа блоков $\mathtt{k}$ определяется следующими соотношениями. Верхняя граница $\mathtt{k} < \min m, 0.5(n+1)$. Нижняя граница выводится из соотношений:

Если $(-2\mathtt{k}^2+(n+2m+3)\mathtt{k}+\mathfrak{z}-2m-2n-mn)/(m-\mathtt{k}) = \mathtt{k}-1$

1) $3 \leq 0.5(n+m+2-\sqrt{4(\mathfrak{z}-n+1)+(n-m)^2}) < 3$

2) $0.5(n+m+2-\sqrt{4(\mathfrak{z}-n+1)+(n-m)^2}) \geq 3$

Если $(-2\mathtt{k}^2+(n+2m+3)\mathtt{k}+\mathfrak{z}-2m-2n-mn)/(m-\mathtt{k}) = 2$

1) $3 \leq 0.25(n+2m+5-\sqrt{4(2\mathfrak{z}-3n+4)+(n-2m+3)^2})$

2) $0.25(n+2m+5-\sqrt{4(2\mathfrak{z}-3n+4)+(n-2m+3)^2}) < 3$

При этом параметры матрицы подчиняются следующим соотношениям: $m \geq 4$, $n \geq 4$, $n+m-1 \leq \mathfrak{z} < 0.5mn$. Исследуем полученные нижние границы. Первая нижняя граница $0.5(n+m+2-\sqrt{4(\mathfrak{z}-n+1)+(n-m)^2})$ всегда больше второй нижней границы $0.25(n+2m+5-\sqrt{4(2\mathfrak{z}-3n+4)+(n-2m+3)^2})$, значит для нижней границы $\mathtt{k}$ возможны только два варианта:

1) $3 \leq 0.25(n+2m+5-\sqrt{4(2\mathfrak{z}-3n+4)+(n-2m+3)^2})$

2) $0.25(n+2m+5-\sqrt{4(2\mathfrak{z}-3n+4)+(n-2m+3)^2}) \leq 3$

Исследуем полученные значения относительно верхней границы. Для того, чтобы существовали решения, необходимо, чтобы верхняя граница не была меньше, чем нижняя. Значит БД–структура существует только в том случае, когда:

1) $3 \leq 0.25(n+2m+5-\sqrt{4(2\mathfrak{z}-3n+4)+(n-2m+3)^2}) \leq min(m, 0.5(n+1))$

2) $0.25(n+2m+5-\sqrt{4(2\mathfrak{z}-3n+4)+(n-2m+3)^2}) \leq 3 \leq min(m, 0.5(n+1))$

Таким образом, для того, чтобы в матрице общего вида можно было выделить БД–структуру, её параметры должны удовлетворять соотношению $m \geq 4$, $n \geq 4$, $n+m-1 \leq \mathfrak{z} < 0.5mn$. При этом число блоков БД–структуры определяется следующим образом:

1) $3 \leq 0.25(n+2m+5-\sqrt{4(2\mathfrak{z}-3n+4)+(n-2m+3)^2}) \leq min(m, 0.5(n+1))$

2) $0.25(n+2m+5-\sqrt{4(2\mathfrak{z}-3n+4)+(n-2m+3)^2}) \leq 3 \leq min(m, 0.5(n+1))$

Теорема доказана.

Теперь сформулируем необходимое условие выделяемости БД–структуры для очень широкой матрицы.

Теорема 1.16. Для того, чтобы БД–структуру можно было выделить для очень широкой матрицы, необходимо, чтобы параметры матрицы удовлетворяли соотношениям: $4 \leq m \leq 0.5n$, $n \geq 8$, $2n-m+1 \leq \mathfrak{z} < 0.5mn$. При этом число блоков ограничивается $0.25(n+2m+5-\sqrt{4(2\mathfrak{z}-3n+4)+(n-2m+3)^2}) \leq \mathtt{k} < m$.

Доказательство. Согласно классификации матриц 1.1, для очень широких матриц имеет место соотношение $3 < m < 0.5(n+1)$. Значит согласно лемме 1.5 число блоков БД–структуры определяется следующим образом:

1) $3 \leq 0.25(n+2m+5-\sqrt{4(2\mathfrak{z}-3n+4)+(n-2m+3)^2}) \leq m$

2) $0.25(n+2m+5-\sqrt{4(2\mathfrak{z}-3n+4)+(n-2m+3)^2}) \leq 3 \leq m$

С учётом особенностей классификации параметры матрицы удовлетворяют условиям: $m \geq 4$, $n \geq 2m$, $n + m - 1 \leq \mathfrak{z} < 0.5mn$. Уточним условия для параметров матрицы. Решим систему неравенств для каждого из случаев:

1)

$$\begin{cases} 3 \leq 0.25(n + 2m + 5 - \sqrt{4(2\mathfrak{z} - 3n + 4) + (n - 2m + 3)^2}) < m, \\ n + m - 1 \leq \mathfrak{z} < 0.5mn, \\ m \geq 4, \\ n \geq 2m. \end{cases}$$

Решения данной системы:

— $m = 4$, $n = 8$, $13 \leq \mathfrak{z} \leq 15$

— $4 \leq m \leq 0.5n$, $n \geq 9$, $2n - m + 1 \leq \mathfrak{z} < 0.5mn$

2)

$$\begin{cases} 3 \geq 0.25(n + 2m + 5 - \sqrt{4(2\mathfrak{z} - 3n + 4) + (n - 2m + 3)^2}), \\ 0.25(n + 2m + 5 - \sqrt{4(2\mathfrak{z} - 3n + 4) + (n - 2m + 3)^2}) \leq m, \\ n + m - 1 \leq \mathfrak{z} < 0.5mn, \\ m \geq 4, \\ n \geq 2m. \end{cases}$$

Данная система не имеет решений

Таким образом, чтобы для очень широкой матрицы выделить БД–структуру, её параметры должны удовлетворять соотношениям:

— $m = 4$, $n = 8$, $13 \leq \mathfrak{z} \leq 15$

— $4 \leq m \leq 0.5n$, $n \geq 9$, $2n - m + 1 \leq \mathfrak{z} < 0.5mn$

При этом число блоков ограничивается $0.25(n + 2m + 5 - \sqrt{4(2\mathfrak{z} - 3n + 4) + (n - 2m + 3)^2}) \leq \mathtt{k} < m$. Теорема доказана.

Аналогично доказывается лемма 1.6 о числе блоков в БД–структуре для для классов матриц, не относящихся к очень широким.

Лемма 1.6. Чтобы для матрицы, не относящейся к классу очень широких, выделить БД–структуру, её параметры должны удовлетворять соотношениям

— $m \geq 5$, $n = 4$, $m + 3 \leq \mathfrak{z} \leq 2m - 2$

— $m \geq 4$, $5 \leq n \leq 2m - 1$, $n + m - 1 \leq \mathfrak{z} < 0.5mn$

— $m \geq 4$, $n = 4$, $\mathfrak{z} = 2m - 1$

При этом число блоков ограничивается $0.25(n + 2m + 5 - \sqrt{4(2\mathfrak{z} - 3n + 4) + (n - 2m + 3)^2}) \leq \mathtt{k} < 0.5(n + 1)$.

Далее сформулируем необходимое условие выделяемости БД–структуры для широкой матрицы.

Теорема 1.17. Для того, чтобы БД–структуру можно было выделить для широкой матрицы, необходимо, чтобы параметры матрицы удовлетворяли соотношениям: $m \geq 4$, $m < n \leq 2m - 1$, $n + m - 1 \leq \mathfrak{z} < 0.5mn$.

Доказательство. Согласно классификации матриц 1.1, для широких матриц имеют место соотношения $0.5(n + 1) \leq m < n$, $n > 3$, $m > 3$. Значит согласно лемме 1.6 число блоков в БД–структуре для широкой матрицы будет следующим: $0.25(n + 2m + 5 - \sqrt{4(2\mathfrak{z} - 3n + 4) + (n - 2m + 3)^2}) \leq \mathtt{k} < 0.5(n + 1)$, если

— $m \geq 5$, $n = 4$, $m + 3 \leq \mathfrak{z} \leq 2m - 2$

— $m \geq 4$, $5 \leq n \leq 2m - 1$, $n + m - 1 \leq \mathfrak{z} < 0.5mn$

$\mathtt{k} = 3$, если $m \geq 4$, $n = 4$, $\mathfrak{z} = 2m - 1$.

Для условий $0.5(n + 1) \leq m < n$, $n > 3$, $m > 3$ подходит только условие $m \geq 4$, $5 \leq n \leq 2m - 1$, $n + m - 1 \leq \mathfrak{z} < 0.5mn$. Решим соответствующую систему неравенств с условиями для широких матриц

$$\begin{cases} n + m - 1 \leq \mathfrak{z} < 0.5mn, \\ 5 \leq n \leq 2m - 1, \\ m \geq 4, \\ 0.5(n + 1) \leq m < n. \end{cases}$$

Решение данной системы: $m \geq 4$, $m < n \leq 2m - 1$, $n + m - 1 \leq \mathfrak{z} < 0.5mn$

Таким образом, для того, чтобы можно было выделить БД–структуру в широкой матрице, её параметры должны быть: $m \geq 4$, $m < n \leq 2m - 1$, $n + m - 1 \leq \mathfrak{z} < 0.5mn$

При этом число блоков ограничивается следующим образом: $0.25(n + 2m + 5 - \sqrt{4(2\mathfrak{z} - 3n + 4) + (n - 2m + 3)^2}) \leq \mathtt{k} < 0.5(n + 1)$. Теорема доказана.

Аналогично доказываются теоремы 1.18, 1.19, в которых формулируется необходимые условия выделяемости БД–структуры для квадратной и узкой матрицы. Вначале рассмотрим случай квадратной матрицы.

Теорема 1.18. Для того, чтобы БД–структуру можно было выделить для квадратной матрицы, необходимо, чтобы параметры матрицы: $n = m \geq 4$, $2n - 1 \leq \mathfrak{z} < 0.5n^2$.

Теперь рассмотрим случай узкой матрицы.

Теорема 1.19. Для того, чтобы БД–структуру можно было выделить для узкой матрицы, необходимо, чтобы параметры матрицы удовлетворяли соотношениям:

— $n = 4$, $m \geq 5$ $m + 3 \leq \mathfrak{z} \leq 2m - 1$

— $m \geq 6$, $5 \leq n < m$, $m + n - 1 \leq \mathfrak{z} < 0.5mn$

Исследуем степень БД–структуры относительно заданных параметров. Следующая лемма определяет вспомогательные соотношения для определения степени в БД–структуре.

Лемма 1.7. Степень БД–структуры определяется следующим образом:

1) $\rho \leq \mathtt{k} - 1$, если выполняется $\mathfrak{z} \geq \mathtt{k}^2 - (n + m + 2)\mathtt{k} + m + 2n + mn$

2) $\rho \leq (-2\mathtt{k}^2 + (n + 2m + 3)\mathtt{k} + \mathfrak{z} - 2m - 2n - mn)/(m - \mathtt{k})$, если выполняется $\mathfrak{z} < \mathtt{k}^2 - (n + m + 2)\mathtt{k} + m + 2n + mn$

Доказательство. В соответствии с теоремой 1.1 получим следующую систему неравенств:

$$\begin{cases} \rho \leq (-2\mathtt{k}^2 + (n + 2m + 3)\mathtt{k} + \mathfrak{z} - 2m - 2n - mn)/(m - \mathtt{k}), \\ 2 \leq \rho \leq \mathtt{k} - 1, \\ 3 \leq \mathtt{k} < \min(m, 0.5(n + 1)). \end{cases}$$

В данной системе верхняя граница ρ определяется двумя способами: $\rho \leq (-2\mathtt{k}^2 + (n + 2m + 3)\mathtt{k} + \mathfrak{z} - 2m - 2n - mn)/(m - \mathtt{k})$ и $\rho \leq \mathtt{k} - 1$. Значит имеют место два случая:

— $2 \leq \rho \leq \mathtt{k} - 1 \leq (-2\mathtt{k}^2 + (n + 2m + 3)\mathtt{k} + \mathfrak{z} - 2m - 2n - mn)/(m - \mathtt{k})$

— $2 \leq \rho \leq (-2\mathtt{k}^2 + (n + 2m + 3)\mathtt{k} + \mathfrak{z} - 2m - 2n - mn)/(m - \mathtt{k}) \leq \mathtt{k} - 1$

Рассмотрим первый случай. Для этого решим неравенство $\mathtt{k} - 1 \leq (-2\mathtt{k}^2 + (n + 2m + 3)\mathtt{k} + \mathfrak{z} - 2m - 2n - mn)/(m - \mathtt{k})$.

Так как $(m - \mathtt{k})$ всегда положительно согласно $\mathtt{k} < m$, имеет место

$(\mathtt{k} - 1)(m - \mathtt{k}) \leq -2\mathtt{k}^2 + (n + 2m + 3)\mathtt{k} + \mathfrak{z} - 2m - 2n - mn$

$\mathfrak{z} \geq \mathtt{k}^2 - (n + m + 2)\mathtt{k} + m + 2n + mn$

Рассмотрим второй случай. Для этого решим неравенство $\mathtt{k} - 1 > (-2\mathtt{k}^2 + (n + 2m + 3)\mathtt{k} + \mathfrak{z} - 2m - 2n - mn)/(m - \mathtt{k})$.

Так как $(m - \mathtt{k})$ всегда положительно согласно $3 \leq \mathtt{k} < m$, имеет место

$(\mathtt{k} - 1)(m - \mathtt{k}) > -2\mathtt{k}^2 + (n + 2m + 3)\mathtt{k} + \mathfrak{z} - 2m - 2n - mn$

$\mathfrak{z} < \mathtt{k}^2 - (n + m + 2)\mathtt{k} + m + 2n + mn$ Теорема доказана.

Рассмотрим каждое из полученных соотношений в лемме 1.7. Для этого сформулируем следующую лемму.

Лемма 1.8. Для того, чтобы верхняя граница степени БД–структуры была $\rho \leq \mathtt{k} - 1$ необходимо, чтобы $0.5(m + n + 2 - \sqrt{(n - m)^2 + 4(\mathfrak{z} - n + 1)}) \leq \mathtt{k} < \min(m, 0.5(n + 1))$. Для того, чтобы верхняя граница степени БД–структуры была $\rho \leq (-2\mathtt{k}^2 + (n + 2m + 3)\mathtt{k} + \mathfrak{z} - 2m - 2n - mn)/(m - \mathtt{k})$ необходимо, чтобы $0.25(n + 2m + 5 - \sqrt{4(2\mathfrak{z} - 3n + 4) + (n - 2m + 3)^2}) \leq \mathtt{k} < 0.5(m + n + 2 - \sqrt{(n - m)^2 + 4(\mathfrak{z} - n + 1)}))$. При этом параметры матрицы:

1) $4 \leq n \leq 7$,$m \geq 4$,$m + n - 1 \leq \mathfrak{z} < nm/2$

2) $n \geq 8$,$4 \leq m < 0.5(n + 1)$, $2n - m \leq \mathfrak{z} < 0.5mn$

3) $n \geq 8$,$m \geq 0.5(n + 1)$, $n + m - 1 \leq \mathfrak{z} < 0.5mn$

Доказательство. Согласно лемме 1.5 верхняя граница $\mathtt{k}$ всегда $\min(m, 0.5(n + 1))$. Нижняя граница для $\mathtt{k}$ — $0.25(n + 2m + 5 - \sqrt{4(2\mathfrak{z} - 3n + 4) + (n - 2m + 3)^2}) \leq 0.5(m + n + 2 - \sqrt{(n - m)^2 + 4(\mathfrak{z} - n + 1)})$. Рассмотрим как $0.25(n + 2m + 5 - \sqrt{4(2\mathfrak{z} - 3n + 4) + (n - 2m + 3)^2}) \leq$

$0.5(m + n + 2 - \sqrt{(n-m)^2 + 4(\mathfrak{z} - n + 1)})$ влияет на параметры исходной матрицы. Решим систему:

$$\begin{cases} n + m - 1 \le \mathfrak{z} < 0.5mn, \\ n > 3, \\ m > 3, \\ 0.25(n + 2m + 5 - \sqrt{4(2\mathfrak{z} - 3n + 4) + (n - 2m + 3)^2}) \le 0.5(m + n + 2 - \sqrt{(n-m) \end{cases}$$

Решения:

1) $4 \le n \le 7, m \ge 4, m + n - 1 \le \mathfrak{z} < nm/2$

2) $n \ge 8, 4 \le m < 0.5(n + 1)$, $2n - m \le \mathfrak{z} < 0.5mn$

3) $n \ge 8, m \ge 0.5(n + 1)$, $n + m - 1 \le \mathfrak{z} < 0.5mn$

Рассмотрим первый случай. Согласно лемме 1.7, при $\rho \le \mathtt{k} - 1$ выполняется $\mathfrak{z} \ge \mathtt{k}^2 - (n + m + 2)\mathtt{k} + m + 2n + mn$. Решим данное неравенство относительно $\mathtt{k}$. Решениями данного неравенства являются:

— $\mathtt{k} = 0.5(m + n + 2)$ при $\mathfrak{z} = 0.25(2mn - m^2 - (n - 2)^2)$ Данное равенство не имеет решений

— $0.5(m + n + 2 - \sqrt{(n-m)^2 + 4(\mathfrak{z} - n + 1)}) \le \mathtt{k} \le 0.5(m + n + 2 + \sqrt{(n-m)^2 + 4(\mathfrak{z} - n + 1)})$ при $\mathfrak{z} > 0.25(2mn - m^2 - (n - 2)^2)$

Таким образом выполняется $0.5(m + n + 2 - \sqrt{(n-m)^2 + 4(\mathfrak{z} - n + 1)}) < \min(m, 0.5(n + 1))$ и $0.5(m + n + 2 + \sqrt{(n-m)^2 + 4(\mathfrak{z} - n + 1)}) > \min(m, 0.5(n + 1))$. Нижняя граница для $\mathtt{k}$ — $0.25(n + 2m + 5 - \sqrt{4(2\mathfrak{z} - 3n + 4) + (n - 2m + 3)^2}) \le 0.5(m + n + 2 - \sqrt{(n-m)^2 + 4(\mathfrak{z} - n + 1)})$. Поскольку $\mathtt{k} \ge 0.5(m + n + 2 - \sqrt{(n-m)^2 + 4(\mathfrak{z} - n + 1)})$, верхняя граница степени БД–структуры $\rho \le \mathtt{k} - 1$ на промежутке $\mathtt{k} \in [0.5(m + n + 2 - \sqrt{(n-m)^2 + 4(\mathfrak{z} - n + 1)}); \min(m, 0.5(n + 1))]$. При этом параметры матрицы:

1) $4 \le n \le 7, m \ge 4, m + n - 1 \le \mathfrak{z} < nm/2$

2) $n \ge 8, 4 \le m < 0.5(n + 1)$, $2n - m \le \mathfrak{z} < 0.5mn$

3) $n \ge 8, m \ge 0.5(n + 1)$, $n + m - 1 \le \mathfrak{z} < 0.5mn$

Рассмотрим второй случай. Согласно лемме 1.7, при $\rho \leq (-2\mathtt{k}^2+(n+2m+3)\mathtt{k}+\mathfrak{z}-2m-2n-mn)/(m-\mathtt{k})$ выполняется $\mathfrak{z} < \mathtt{k}^2-(n+m+2)\mathtt{k}+m+2n+mn$. Решим данное неравенство относительно $\mathtt{k}$. Решениями данного неравенства являются:

— $\mathtt{k} < 0.5(m+n+2)$ при $\mathfrak{z} = 0.25(2mn-m^2-(n-2)^2)$

— $\mathtt{k} > 0.5(m+n+2)$ при $\mathfrak{z} = 0.25(2mn-m^2-(n-2)^2)$ Нет решений

— $\mathtt{k} > 0.5(m+n+2+\sqrt{(n-m)^2+4(\mathfrak{z}-n+1)})$ при $\mathfrak{z} > 0.25(2mn-m^2-(n-2)^2)$ Нет решений

— $\mathtt{k} < 0.5(m+n+2-\sqrt{(n-m)^2+4(\mathfrak{z}-n+1)})$ при $\mathfrak{z} > 0.25(2mn-m^2-(n-2)^2)$

Таким образом, итоговое решение: $\mathtt{k} < 0.5(m+n+2-\sqrt{(n-m)^2+4(\mathfrak{z}-n+1)})$ при $\mathfrak{z} \geq 0.25(2mn-m^2-(n-2)^2)$. Поскольку $\mathtt{k} < 0.5(m+n+2-\sqrt{(n-m)^2+4(\mathfrak{z}-n+1)})$, верхняя граница степени БД–структуры $\rho \leq (-2\mathtt{k}^2+(n+2m+3)\mathtt{k}+\mathfrak{z}-2m-2n-mn)/(m-\mathtt{k})$ на промежутке $\mathtt{k} \in [0.25(n+2m+5-\sqrt{4(2\mathfrak{z}-3n+4)+(n-2m+3)^2}); 0.5(m+n+2-\sqrt{(n-m)^2+4(\mathfrak{z}-n+1)})]$. Теорема доказана.

Рассмотрим, как меняются условия леммы 1.8 относительно различных матриц. Вначале сформулируем лемму для матриц, относящихся к классу очень широких.

Лемма 1.9. Параметры матрицы, относящейся к классу очень широких: $m \geq 4$, $n \geq 2m$, $2n-m+1 \leq \mathfrak{z} < 0.5mn$.

Доказательство. Согласно лемме 1.8 параметры матрицы:

1) $4 \leq n \leq 7$,$m \geq 4$,$m+n-1 \leq \mathfrak{z} < nm/2$

2) $n \geq 8$,$4 \leq m < 0.5(n+1)$, $2n-m \leq \mathfrak{z} < 0.5mn$

3) $n \geq 8$,$m \geq 0.5(n+1)$, $n+m-1 \leq \mathfrak{z} < 0.5mn$

Согласно теореме 1.16, для очень широкой матрицы подходят только 2 условие, поскольку параметры очень широкой матрицы: $4 \leq m \leq 0.5n$, $n \geq 8$, $2n-m+$

$1 \leq \mathfrak{z} < 0.5mn$.

$$\begin{cases} 2n - m \leq \mathfrak{z} < 0.5mn, \\ 2n - m + 1 \leq \mathfrak{z} < 0.5mn, \\ n \geq 8, \\ 4 \leq m \leq 0.5n, \\ 4 \leq m < 0.5(n+1). \end{cases}$$

Решение данной системы: $m \geq 4$, $n \geq 2m$, $2n - m + 1 \leq \mathfrak{z} < 0.5mn$. Таким образом, для очень широкой матрицы параметры: $m \geq 4$, $n \geq 2m$, $2n - m + 1 \leq \mathfrak{z} < 0.5mn$. Теорема доказана.

Лемма 1.10 доказывается аналогично и формулируется относительно матриц, не относящихся к классу очень широких.

Лемма 1.10. Параметры матрицы, не относящейся к классу очень широких: $m \geq 4$, $4 \leq n \leq 7$, $m + n - 1 \leq \mathfrak{z} < 0.5mn$ или $m \geq 5$, $8 \leq n \leq 2m - 1$, $m + n - 1 \leq \mathfrak{z} < 0.5mn$.

Далее сформулируем лемму относительно широких матриц.

Лемма 1.11. Параметры широкой матрицы:

— $4 \leq m \leq 6$, $m + 1 \leq n \leq 7$, $n + m - 1 \leq \mathfrak{z} < 0.5mn$

— $5 \leq m \leq 7$, $8 \leq n \leq 2m - 1$, $n + m - 1 \leq \mathfrak{z} < 0.5mn$

— $m \geq 8$, $m + 1 \leq n \leq 2m - 1$, $n + m - 1 \leq \mathfrak{z} < 0.5mn$

Доказательство. Согласно лемме 1.10 параметры матрицы $m \geq 4$, $4 \leq n \leq 7$, $m + n - 1 \leq \mathfrak{z} < 0.5mn$ или $m \geq 5$, $8 \leq n \leq 2m - 1$, $m + n - 1 \leq \mathfrak{z} < 0.5mn$. Согласно теореме 1.17 необходимо, чтобы параметры матрицы удовлетворяли соотношениям: $m \geq 4$, $m < n \leq 2m - 1$, $n + m - 1 \leq \mathfrak{z} < 0.5mn$. Решим системы относительно данных условий.

1

$$\begin{cases} m + n - 1 \leq \mathfrak{z} < 0.5mn, \\ 4 \leq n \leq 7, \\ m < n \leq 2m - 1, \\ m \geq 4. \end{cases}$$

Решение данной системы: $4 \leq m \leq 6, m+1 \leq n \leq 7, n+m-1 \leq \mathfrak{z} < 0.5mn$

2

$$\begin{cases} n + m - 1 \leq \mathfrak{z} < 0.5mn, \\ 8 \leq n \leq 2m - 1, \\ m < n \leq 2m - 1, \\ m \geq 5. \end{cases}$$

Решения данной системы:

— $5 \leq m \leq 7, 8 \leq n \leq 2m - 1, n + m - 1 \leq \mathfrak{z} < 0.5mn$

— $m \geq 8, m + 1 \leq n \leq 2m - 1, n + m - 1 \leq \mathfrak{z} < 0.5mn$

Таким образом параметры широкой матрицы:

— $4 \leq m \leq 6, m + 1 \leq n \leq 7, n + m - 1 \leq \mathfrak{z} < 0.5mn$

— $5 \leq m \leq 7, 8 \leq n \leq 2m - 1, n + m - 1 \leq \mathfrak{z} < 0.5mn$

— $m \geq 8, m + 1 \leq n \leq 2m - 1, n + m - 1 \leq \mathfrak{z} < 0.5mn$

Теорема доказана.

Леммы 1.12, 1.13 доказывается аналогично и формулируются относительно квадратных и узких матриц. Вначале рассмотрим случай квадратной матрицы.

Лемма 1.12. Параметры квадратной матрицы: $m = n$, $n \geq 4$, $2n - 1 \leq \mathfrak{z} < 0.5n^2$

Теперь рассмотрим случай узкой матрицы.

Лемма 1.13. Параметры узкой матрицы:

— $6 \leq m \leq 7, 5 \leq n \leq m - 1\ m + n - 1 \leq \mathfrak{z} < 0.5mn$

— $m \geq 8, 5 \leq n \leq 7, m + n - 1 \leq \mathfrak{z} < 0.5mn$

— $m \geq 9, 8 \leq n \leq m - 1, m + n - 1 \leq \mathfrak{z} < 0.5mn$

На основе доказанных лемм сформулируем финальную теорему о составе БД–структуры.

Теорема 1.20. Степень БД–структуры подчиняется соотношению $2 \le \rho \le \mathtt{k} - 1$, если число блоков $0.5(m + n + 2 - \sqrt{(n-m)^2 + 4(\mathfrak{z} - n + 1)}) \le \mathtt{k} < \min(m, 0.5(n+1))$.

Степень БД–структуры подчиняется соотношению $2 \le \rho \le (-2\mathtt{k}^2 + (n + 2m + 3)\mathtt{k} + \mathfrak{z} - 2m - 2n - mn)/(m - \mathtt{k})$, если число блоков $0.25(n + 2m + 5 - \sqrt{4(2\mathfrak{z} - 3n + 4) + (n - 2m + 3)^2}) \le \mathtt{k} < 0.5(m + n + 2 - \sqrt{(n-m)^2 + 4(\mathfrak{z} - n + 1)}))$.

При этом параметры $n, m, \mathfrak{z}$:

1. Для очень широких матриц $m \ge 4$, $n \ge 2m$, $2n - m + 1 \le \mathfrak{z} < 0.5mn$
2. Для широких матриц
   - $4 \le m \le 6$, $m + 1 \le n \le 7$, $n + m - 1 \le \mathfrak{z} < 0.5mn$
   - $5 \le m \le 7$, $8 \le n \le 2m - 1$, $n + m - 1 \le \mathfrak{z} < 0.5mn$
   - $m \ge 8$, $m + 1 \le n \le 2m - 1$, $n + m - 1 \le \mathfrak{z} < 0.5mn$
3. Для квадратных матриц $m = n$, $n \ge 4$, $2n - 1 \le \mathfrak{z} < 0.5n^2$
4. Для узких матриц
   - $6 \le m \le 7$, $5 \le n \le m - 1$ $m + n - 1 \le \mathfrak{z} < 0.5mn$
   - $m \ge 8$, $5 \le n \le 7$, $m + n - 1 \le \mathfrak{z} < 0.5mn$
   - $m \ge 9$, $8 \le n \le m - 1$, $m + n - 1 \le \mathfrak{z} < 0.5mn$

1.3 Выделение квазиблочной структуры

Далее исследуются алгоритмы выделения БЛ– и БД–структур. Принцип объединения вершин в блоки был предложен Ю.Ю. Финкельштейном [82] для выделения БЛ–структуры в разреженной матрице. Представлен модифицированный алгоритм, также предложена модификация алгоритма, направленная на уменьшение количества блоков и сепараторов. Показано как с помощью данного подхода осуществляется выделение БД–структуры матриц.

Разреженные матрицы будут описываться с помощью структурных графов. Вершины структурного графа определяются его видом: если структурный

граф является графом взаимосвязей, то его вершины соответствуют отдельным столбцам, а если структурный граф — граф взаимосвязей блоков, то его вершины соответствуют подмножествам столбцов (рис. 2.1).

Более адекватным является представление матрицы в виде гиперграфа: множество вершин гиперграфа H соответствует множеству вершин X графа взаимосвязей матрицы, а гиперребра гиперграфа образуют подмножества взаимосвязанных вершин. То есть это такое обобщение графа взаимосвязей, в котором каждым ребром могут соединяться любые подмножества вершин. На рисунке 1.8 изображён гиперграф, множеством вершин которого является множество веершин графа взаимосвязей $\{x_1, \ldots, x_7\}$, а множество гиперрёбер — $\{C_1, C_2, C_3, C_4\}$ соответствует строкам матрицы.

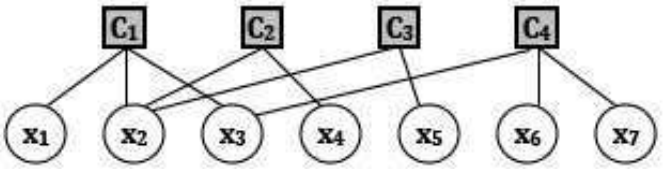


Рисунок 1.8: Гиперграф

Также используется двойственный граф для представления структуры матрицы, то есть граф, вершины которого соответствуют гиперребрам гиперграфа. Пара таких вершин связаны ребром, если соответствующие гиперрёбра смежны относительно X (см. рис. 1.9).

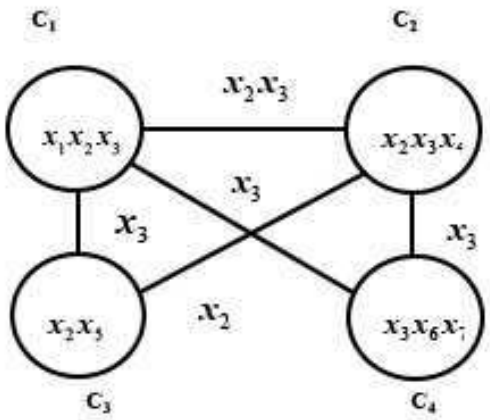


Рисунок 1.9: Двойственный граф

Алгоритм [82] позволяет выделять из двойственного графа БЛ–структуру. Он строится следующим образом. Пусть $A_{m\times n}$ – разреженная матрица, p – ин-

декс вершины $x_p, p \in \{1, 2, \ldots, n\}$. Далее необходимо выделить окрестности вершины x_p последовательно возрастающих порядков, т.е.

$$S_0^{'}(x_p) = \{p\}, U_{r+1}^{'} = U(S_r^{'}(x_p)), r \geq 0; \tag{1.5}$$

$$S_{r+1}^{'} = S(U_r^{'}(x_p)), r \geq 1 \tag{1.6}$$

Выделение окрестностей заканчивается, когда при $r = k$ выполняется условие $S_{k+1}^{'}(x_p) = S_k^{'}(x_p)$. После этого строится разбиение множества индексов строк, входящих в блоки, по следующему принципу:

$$U_r = \begin{cases} U_1^{'}(x_p), & r = 1, \\ U_r^{'}(x_p) \backslash U_{r-1}^{'}(x_p), & 1 < r < k, \\ U_k^{'}(x_p) \bigcup U_{k+1}^{'}(x_p) \backslash U_{k-1}^{'}(x_p), & r = k. \end{cases}$$

Затем находятся следующие множества индексов столбцов:

$$S_{r,r+1} = S(U_r(x_p)) \bigcap S(U_{r+1}(x_p)),$$

$$S_r = \begin{cases} S\left(U_1\left(x_p\right)\right) \backslash S_{12}, & r = 1, \\ S\left(U_r\left(x_p\right)\right) \backslash \left(S_{r-1,r} \cup S_{r,r+1}\right), & 1 < r < k, \\ S\left(U_k\left(x_p\right)\right) \backslash S_{k-1,k}, & r = k. \end{cases}$$

Описанный алгоритм находит БЛ–структуру для разреженной матрицы. Заметим, что БЛ–структуры могут отличаться друг от друга, если первым будет выбран другой столбец в матрице. Рассмотрим пример. Пусть дана система неравенств следующего вида:

$$\begin{cases} 2x_3 + x_5 + 2x_{12} + 4x_{13} \leq 6 \\ 2x_1 + 3x_2 + 3x_5 + 2x_{11} \leq 4 \\ x_6 + 4x_7 + 8x_{12} + 3x_{14} \leq 8 \\ 3x_4 + 3x_{10} + x_{14} \leq 4 \\ x_6 + 2x_8 + 2x_9 \leq 2 \\ x_j \in \{0, 1\}, j = 1, 2, \ldots, 14. \end{cases} \tag{1.7}$$

Матрица ограничений предложенной задачи:

$$\begin{pmatrix} 0 & 0 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 4 & 1 & 0 & 0 & 0 \\ 2 & 0 & 3 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 & 0 & 0 & 0 & 1 & 3 & 4 \\ 0 & 3 & 0 & 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Зафиксируем, например, вершину x_1. Рассмотрим возрастающие окрестности этой вершины. Очевидно, что окрестностью нулевого порядка будет индекс выбранной вершины $S_0^{'} = \{1\}$. Множество индексов строк $U_1^{'} = U(S_0^{'}(x_1))$. Поскольку данная переменная находится во втором ограничении, $U_1^{'} = U(1) = \{2\}$. Второе ограничение — это $2x_1 + 3x_2 + 3x_5 + 2x_{11} \leq 4$, значит окрестностью первого порядка будет множество индексов переменных $S_1^{'} = \{2; 5; 11\}$. При этом множество индексов ограничений $U_2^{'} = U(S_1^{'}(x_2; x_5; x_{11}))$. Это соответствует первому ограничению, поскольку переменные $\{x_2; x_{11}\}$ являются свободными переменными, а $\{x_5\}$ содержится в неравенстве $2x_3 + x_5 + 2x_{12} + 4x_{13} \leq 6$. Значит $U_2^{'} = U(2; 5; 11) = \{1\}$. Таким образом, окрестностью второго порядка для x_1 будет множество $S_2^{'} = \{3; 12; 13\}$. Построим множество индексов ограничений третьего порядка $U_3^{'} = U(S_2^{'}(x_3; x_{12}; x_{13}))$. Рассмотрим каждую переменную множества $S_2^{'}$: $\{x_3; x_{13}\}$ — свободные переменные, а а $\{x_{12}\}$ содержится в неравенстве $x_6 + 4x_7 + 8x_{12} + 3x_{14} \leq 8$. Это третье неравенство. Значит $U_3^{'} = U(3; 12; 13) = \{3\}$, а окрестностью третьего порядка будет $S_3^{'} = \{6; 7; 14\}$. Построим $U_4^{'} = U(S_3^{'}(x_6; x_7; x_{14}))$. Две переменные, соответствующие данному множеству индексов, а именно — x_6 и x_{14}, входят в ограничения $x_6 + 2x_8 + 2x_9 \leq 2$ и $3x_4 + 3x_{10} + x_{14} \leq 4$ соответственно. Значит $U_4^{'} = U(6; 7; 14) = \{4; 5\}$, а окрест-

ности четвёртого порядка будет соответствовать множество $S_4^{'}$: $\{4; 8; 9; 10\}$. Переменные с этими индексами не связаны больше ни с какими другими не рассмотренными переменными, поэтому выполняется условие $S_{k+1}^{'}(x_p) = S_k^{'}(x_p)$. Затем строится разбиение множества индексов ограничений, входящих в блоки, следующим образом:

$$U_r = \begin{cases} U_1^{'}(x_1) = \{2\}, & r = 1, \\ U_2^{'}(x_1) \backslash U_1^{'}(x_1) = \{1\}, & r = 2, \\ U_3^{'}(x_1) \backslash U_2^{'}(x_1) = \{3\}, & r = 3, \\ U_4^{'}(x_1) \bigcup U_5^{'}(x_1) \backslash U_3^{'}(x_1) = \{4; 5\}, & r = 4, \end{cases}$$

Затем находятся следующие множества, которые содержат индексы сепараторов БЛ–структуры:

$$S_{12} = S(U_1(x_1)) \bigcap S(U_2(x_1)) = \{5\},$$

$$S_{23} = S(U_2(x_1)) \bigcap S(U_3(x_1)) = \{12\},$$

$$S_{34} = S(U_3(x_1)) \bigcap S(U_4(x_1)) = \{6; 14\},$$

Далее находятся множества, содержащие индексы свободных переменных:

$$S_r = \begin{cases} S\left(U_1\left(x_1\right)\right) \backslash S_{12} = \{2; 11\}, & r = 1, \\ S\left(U_2\left(x_1\right)\right) \setminus \left(S_{23} \cup S_{34}\right) = \{3; 13\}, & r = 2, \\ S\left(U_3\left(x_1\right)\right) \backslash S_{34} = \{4; 8; 9; 10\}, & r = 3. \end{cases}$$

Таким образом, номера окрестностей соответствуют номерам блоков, то есть первый блок соответствует второму ограничению, поскольку $U_1^{'}(x_1) = \{2\}$. В нём содержатся свободные переменные с индексами $S_0 = \{1\}$ и $S_1 = \{2; 11\}$, сепаратор с индексом $S_{12} = \{5\}$. Второй блок будет соответствует первому ограничению и содержит переменные из множеств S_{12}, S_2, S_{23}. Третий блок соответствует третьему ограничению и содержит переменные из множеств S_{23}, S_3, S_{34}. Четвёртый блок соответствует четвёртому и пятому ограничениям и содержит переменные из множеств S_{34}, S_4.

Изменённая матрица ограничений имеет вид:

$$\begin{pmatrix} 2 & 3 & 2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 4 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 8 & 4 & 1 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 3 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 2 & 2 \end{pmatrix}$$

Заметим, что данный алгоритм содержит следующий недостаток. Сепараторы могут оказаться очень большими, то есть состоять из большого числа переменных. Это крайне усложняет работу с матрицей (например, используя локальный блочно–элиминационный алгоритм, который будет подробно описан в четвёртой главе). Поэтому есть смысл добавлять ограничение на размер сепаратора и, если он превышает допустимый, выполнять процедуру слияния блоков. То есть, если сепаратор между между блоками $S_{a,b}$ такой, что его мощность $|S_{a,b}| < s_{max}$, где s_{max} — допустимая величина сепаратора, то новый блок U_{new} будет соответствовать $U_a \bigcup U_b$. Сепараторами этого блока будут множества $S_{a-1,a}$ и $S_{b,b+1}$, а свободными переменными — $S_{new} = S_a \bigcup S_b$.

Далее рассматривается численный эксперимент сравнения основного и модифицированного алгоритма. Целью данного эксперимента является тестирование алгоритма выделения БЛ–структур для разреженных матриц реальной размерности. Опишем, как именно был модифицирован основной алгоритм. Производился поиск вырожденных блоков, то есть таких, для которых $S_r = \emptyset$. При нахождении вырожденных блоков происходило их склеивание, т.е. пустой блок сливался со следующим за ним. Также была добавлена возможность устанавливать пороговую величину сепаратора. Если размер сепаратора превышал заданный порог, то блоки также объединялись.

Алгоритм [82] был реализован на языке программирования C++ и встроен как часть библиотеки LES. Матрицы для эксперимента брались из библиотеки CSP [6]. Выбранные матрицы разбивались на пять групп согласно структуре графа взаимосвязей (рис. 1.10): 'dubois', 'bridge', 'adder', 'pret' и 'grid' , при этом для каждой матрицы строилась БЛ–структура с помощью основного и модифицированного алгоритмов. Результаты численного эксперимента приведены в таблице 1.3.

[6] http://www.dbai.tuwien.ac.at/proj/hypertree

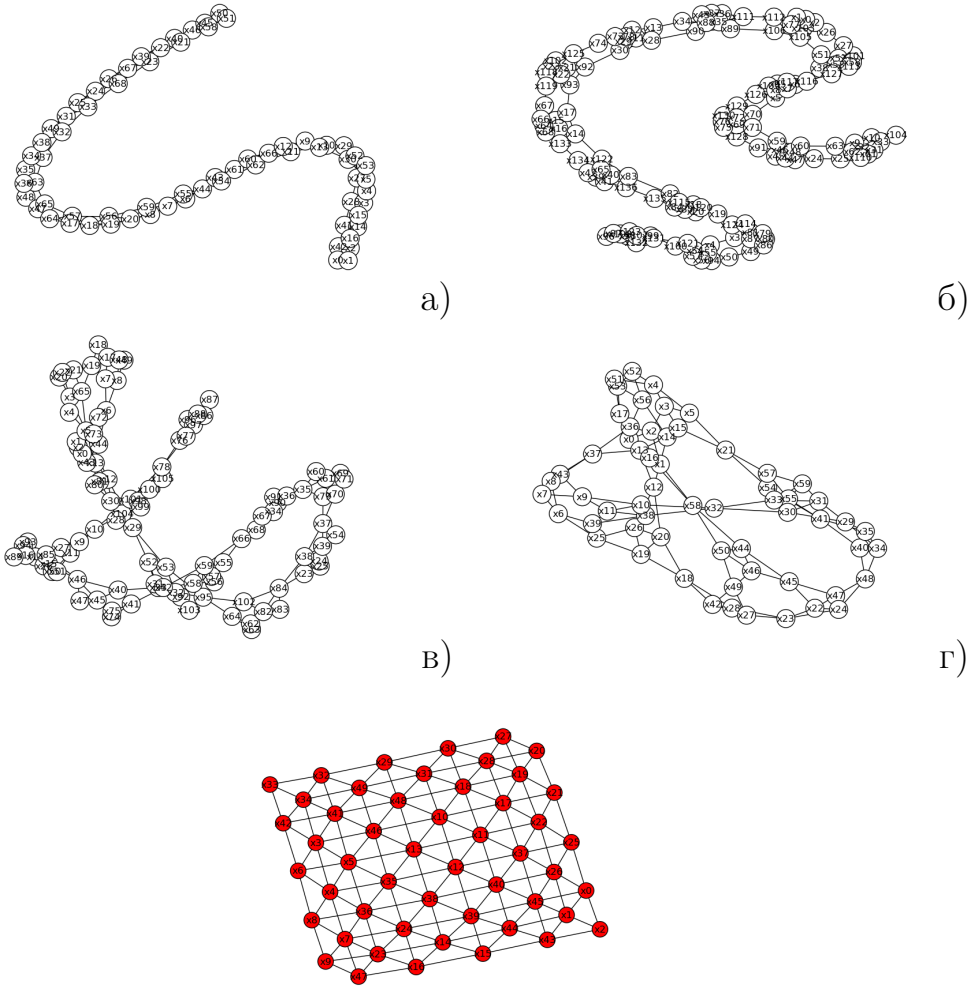


Рисунок 1.10: Примеры тестовых задач из соответствующих групп: а) 'dubois', б) 'bridge', в) 'adder', г) 'pret' и д) 'grid'

Следует отметить, что в качестве первого столбца, используемого алгоритмом, для которого находилась окрестность, брался столбец с номером $p = \{0\}$.

Рассмотрим подробнее работу основного алгоритма на примере системы ограничений с матрицей 'grid2d_10'. Данная матрица состоит из 50–ти столбцов и 50–ти строк. Выделим БЛ–структуру из двойственного графа следующим образом. Зафиксируем столбец с номером 0. $U_0 = \{0, 10, 36\}$ — множество индексов ограничений, в которые входит переменная x_0. Это ограничения B_0, B_{10}, B_{36}:

$$B_0 : x_0 + x_1 + x_2 <= 1$$

$$B_{10} : x_0 + x_{25} + x_{26} <= 1$$

$$B_{36} : x_0 + x_1 + x_{26} + x_{45} <= 1$$

Множеством индексов ограничений следующего порядка $U_1 = \{15, 18, 20, 21, 38, 46\}$ является множество индексов ограничений, в которые входят переменные окрестности x_0, но не входит сама переменная x_0:

$$B_{15} : x_{21} + x_{22} + x_{25} <= 1$$

$$B_{18} : x_{22} + x_{25} + x_{26} + x_{37} <= 1$$

$$B_{20} : x_1 + x_2 + x_{43} <= 1$$

$$B_{21} : x_1 + x_{43} + x_{44} + x_{45} <= 1$$

$$B_{38} : x_{26} + x_{37} + x_{40} + x_{45} <= 1$$

$$B_{46} : x_{39} + x_{40} + x_{44} + x_{45} <= 1$$

У множеств U_0 и U_1 есть общие элементы. Это переменные окрестности x_0 без переменной x_0: $S_0 = \{1, 2, 25, 26, 45\}$.

Аналогичным образом строим множества U более высоких порядков. Таким образом рёбрами гиперграфа будут следующие множества:

$$U_0 = \{0, 10, 36\}$$

$$U_1 = \{15, 18, 20, 21, 38, 46\}$$

$$U_2 = \{6, 7, 16, 19, 31, 39, 40, 41\}$$

$$U_3 = \{3, 4, 5, 8, 11, 26, 27, 32, 33, 49\}$$

$$U_4 = \{2, 14, 23, 28, 30, 35, 37, 43, 44, 47\}$$

$$U_5 = \{1, 9, 12, 22, 24, 34, 45\}$$

$$U_6 = \{17, 25, 29, 42\}$$

$$U_7 = \{13, 48\}$$

Для двойственного графа множества U будут являться гипервершинами, а гиперрёбрами будут множества индексов возрастающих окрестностей S:

$$S_0 = \{1, 2, 25, 26, 45\}$$

$$S_1 = \{21, 22, 37, 39, 40, 43, 44\}$$

$$S_2 = \{11, 12, 14, 15, 17, 19, 20, 24, 38\}$$

$$S_3 = \{7, 10, 13, 16, 18, 23, 27, 28, 35, 36\}$$

$$S_4 = \{4, 5, 8, 9, 30, 31, 46, 47, 48\}$$

$$S_5 = \{3, 6, 29, 41, 49\}$$

$$S_6 = \{32, 34, 42\}$$

Рисунок 1.11: БЛ–структура матрицы

Теперь покажем преимущество модифицированной версии алгоритма над основной на примере системы ограничений для матрицы 'bridge_15'. В данной матрице 137 столбцов и 137 строк. В случае основного и модифицированного

алгоритмов для соответствующей системы ограничений получены следующие множества индексов ограничений 1.2.

Рассмотрим результаты работы программы. Модифицированный алгоритм осуществил слияние блоков таким образом, что множества индексов ограничений:

$$U_0^{new} = U_0^{old} \bigcup U_1^{old}$$

$$U_1^{new} = U_2^{old} \bigcup U_3^{old}$$

$$U_2^{new} = U_4^{old} \bigcup U_5^{old}$$

$$U_3^{new} = U_6^{old} \bigcup U_7^{old}$$

$$U_4^{new} = U_8^{old} \bigcup U_9^{old}$$

$$U_5^{new} = U_{10}^{old} \bigcup U_{11}^{old}$$

$$U_6^{new} = U_{12}^{old} \bigcup U_{13}^{old} \bigcup U_{14}^{old}$$

$$U_7^{new} = U_{15}^{old} \bigcup U_{16}^{old}$$

$$U_8^{new} = U_{17}^{old} \bigcup U_{18}^{old}$$

$$U_9^{new} = U_{19}^{old}$$

$$U_{10}^{new} = U_{20}^{old}$$

Здесь U_j^{new} — множества индексов ограничений, полученные с помощью модифицированной версии алгоритма, а U_j^{old} — основным алгоритмом. При этом множества индексов сепараторов следующие:

$$S_{0,1}^{new} = S_{1,2}^{old}$$

$$S_{1,2}^{new} = S_{3,4}^{old}$$

$$S_{2,3}^{new} = S_{5,6}^{old}$$

$$S_{3,4}^{new} = S_{7,8}^{old}$$

$$S_{4,5}^{new} = S_{9,10}^{old}$$

$$S_{5,6}^{new} = S_{11,12}^{old}$$

Таблица 1.2: Множества индексов ограничений для задачи 'bridge_15', полученные с помощью основного и модифицированного алгоритмов

Множество	Основной алгоритм	Модифицированный алгоритм
U_0	0, 36, 55, 87	0, 36, 55, 57, 74, 87, 103
U_1	57, 74, 103	9, 21, 42, 51, 61, 95, 127, 135
U_2	9, 61, 95, 135	2, 10, 12, 13, 14, 17, 25, 46, 62, 68, 72, 79, 99, 123, 128, 133
U_3	21, 42, 51, 127	4, 33, 34, 37, 43, 45, 48, 59, 65, 78, 82, 93, 98, 102, 111, 112, 113, 121
U_4	13, 14, 17, 25, 46, 62, 68, 72, 79, 123, 128, 133	5, 7, 26, 35, 52, 53, 67, 73, 80, 81, 100, 101, 104, 108, 114, 117, 134, 136
U_5	2, 10, 12, 99	8, 18, 19, 22, 28, 29, 31, 32, 44, 54, 66, 84, 85, 105, 107, 116, 130, 131
U_6	4, 34, 37, 43, 48, 59, 65, 82, 98, 111, 112, 113	3, 6, 11, 15, 16, 27, 30, 39, 56, 60, 64, 69, 71, 75, 77, 88, 94, 96, 109, 115, 118, 124, 125, 129
U_7	33, 45, 78, 93, 102, 121	1, 38, 40, 41, 63, 70, 89, 92, 120
U_8	7, 35, 52, 53, 67, 73, 81, 108, 114, 117, 134, 136	20, 23, 24, 76, 83, 97, 119, 122, 132
U_9	5, 26, 80, 100, 101, 104	47, 50, 106
U_{10}	18, 19, 22, 29, 31, 32, 44, 54, 105, 107, 116, 130	49, 58, 86, 90, 110, 126
U_{11}	8, 28, 66, 84, 85, 131	
U_{12}	3, 11, 15, 27, 30, 39, 77, 88, 94, 96, 109, 118	
U_{13}	16, 56, 60, 71, 124, 129	
U_{14}	6, 64, 69, 75, 115, 125	
U_{15}	40, 63, 92	
U_{16}	1, 38, 41, 70, 89, 120	
U_{17}	20, 76, 83	
U_{18}	23, 24, 97, 119, 122, 132	
U_{19}	47, 50, 106	
U_{20}	49, 58, 86, 90, 110, 126	

$$S^{new}_{6,7} = S^{old}_{14,15}$$

$$S^{new}_{7,8} = S^{old}_{16,17}$$

$$S^{new}_{8,9} = S^{old}_{18,19}$$

$$S^{new}_{9,10} = S^{old}_{19,20}$$

Здесь S^{new}_{ij} — множества индексов сепараторов, полученные с помощью модифицированной версии алгоритма, а S^{old}_{ij} — основным алгоритмом. Рассмотрим соотношение множеств индексов свободных переменных. Множества, полученные основным алгоритмом: $S^{old}_0 = \{0\}$, $S^{old}_4 = \{37, 43, 58, 113\}$, $S^{old}_{13} = \{104\}$, $S^{old}_{20} = \{96, 97, 108, 124\}$. Остальные 16 множеств пусты. При этом множества,полученные модифицированным алгоритмом: $S^{new}_0 = \{0, 1, 2, 6, 27, 28, 34, 51, 77, 89, 103, 111, 128\}$, $S^{new}_1 = \{17, 23, 59, 75, 76, 102, 118, 119, 129, 131\}$, $S^{new}_2 = \{24, 44, 45, 47, 63, 64, 66, 68, 122, 133\}$, $S^{new}_3 = \{3, 10, 18, 31, 33, 39, 41, 42, 50, 54, 55, 61, 81, 82, 87, 104, 110, 114, 115, 120, 121, 135, 136\}$ $S^{new}_4 = \{96, 97, 108, 124\}$. Больше множеств нет. Таким образом, модифицированный алгоритм позволяет решать на 16 подзадач меньше, чем основной.

Анализируя результаты полученные в таблице 1.3, нетрудно заметить эффективность работы модифицированного алгоритма. Так во многих матрицах было существенно уменьшено количество блоков и размер сепараторов, что напрямую влияет на количество решаемых подзадач.

Исследуем подробнее результаты, полученные для некоторых тестовых матриц. Обратим внимание на матрицу 'dubois100'. Для неё число блоков сильно уменьшилось, а размер максимального сепаратора сократился на 2. Для матриц 'pret150_25' и 'pret150_75' сильно сократилось число блоков.

Основным результатом данного эксперимента является тестирование модифицированного алгоритма для выделения БЛ–структур в разреженных матрицах. Произведен сравнительный эксперимент для основной и модифицированной версий алгоритма, показавший существенное уменьшение количества решаемых подзадач для модифицированного алгоритма.

Как показано в [142], переход от БД–структуры к БЛ– путем объединения «слоев» дерева нецелесообразен. В связи с этим представляет интерес выделение БД–структуры в разреженном графе с помощью модификации описанного выше алгоритма.

Таблица 1.3: Результаты декомпозиции задач ДО с помощью основного и модифицированного алгоритма

Задача	Основной алгоритм		Модифицированный алгоритм	
	Кол-во блоков	Максимальный сепаратор	Кол-во блоков	Максимальный сепаратор
adder_15	31	4	16	4
adder_50	61	6	51	6
adder_99	121	6	101	6
dubois23	23	2	23	2
dubois30	30	2	30	2
dubois50	50	2	50	2
dubois100	98	6	2	4
pret60_25	8	10	7	10
pret60_60	8	10	7	10
pret150_25	15	12	11	10
pret150_75	15	12	11	10
grid2d_10	8	10	2	3
grid10	16	10	3	9
grid3d_4	5	12	2	6
bridge_15	21	10	4	10
bridge_50	56	10	5	8
bridge_75	78	9	4	9

В статье [88] для преобразования БЛ–структуры в БД– предложен алгоритм выделения компонент связности. Ниже предложена другая идея модификации основного алгоритма для выделения БД–структуры — МДФ(БД).

Предлагаемая МДФ(БД) состоит из следующих этапов:

Алгоритм 1 Модификация основного алгоритма

Шаг 1. Выделение окрестностей возрастающих порядков.

Шаг 2. Разбиение множества индексов ограничений.

Шаг 3. Нахождение блоков.

Шаг 4. Выделение подблоков.

Первые три шага фактически являются шагами алгоритма [82], рассмотренного выше. Остановимся на четвёртом шаге — выделение подблоков. Полученная с помощью основного алгоритма БЛ–структура $\{U_1, \ldots, U_k\}$ задачи ДО на двойственном графе D имеет вид цепи. Необходимо узнать, возможно ли расщепить блоки на подблоки таким образом, чтобы образовалась БД–структура. Для этого воспользуемся теоремой о составе БД–структуры из параграфа 1.1. Далее, если существует степень БД–структуры, большая двух (для БЛ–структуры), значит БД–структуру указанной степени можно выделить. Перейдём к алгоритму выделения подблоков:

Алгоритм 2 Процедура выделения подблоков в БЛ–структуре

Шаг 1. Исследование сепаратора относительно текущего блока.

Шаг 2. Расщепление сепаратора, если существуют несвязанные ограничения в блоке–потомке.

Шаг 3. Расщепление блока–потомка на подблоки.

Проиллюстрируем этапы МДФ(БД) на примере ограничений задачи ДО следующего вида:

$$\begin{cases} x_4 + 2x_5 + 4x_6 + 2x_7 \leq 6 \\ 3x_1 + 2x_2 + 2x_3 + 3x_4 \leq 4 \\ 8x_7 + 4x_8 + 3x_9 + x_{10} \leq 8 \\ x_9 + 3x_{11} + 3x_{12} \leq 4 \\ x_{10} + 2x_{13} + 2x_{14} \leq 2 \\ x_j \in \{0, 1\}, j = 1, 2, \ldots, 14. \end{cases} \tag{1.8}$$

Введём графовое представление для этой задачи. Построим граф ограничении задачи ДО (1.12) из примера (1.8). Переменные задачи соответствуют вершинам графа, а неравенства определяют наличие рёбер между вершинами. Если вершины находятся в одном неравенстве, они все соединяются рёбрами между собой. То есть, множество вершин — X, а множество рёбер состоит из 24–х элементов. Некоторые из ограничений — (x_1, x_2); (x_1, x_3); (x_2, x_3); (x_4, x_5). Построим гиперграф для задачи ДО 1.13 из примера (1.8). Вершинами гипер-

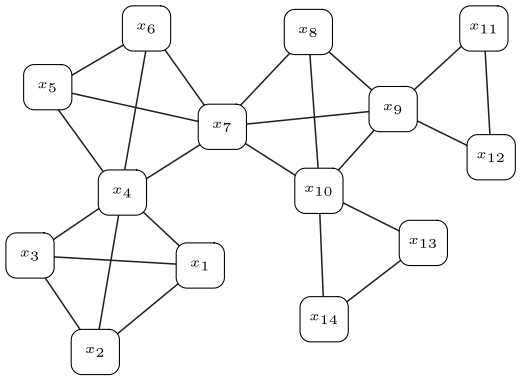


Рисунок 1.12: Графовое представление задачи ДО: граф ограничений

графа будет множество переменных, а гиперрёбра образуют подмножества взаимосвязанных переменных. Таким образом, множество вершин — X, а множество гиперрёбер — B_1, B_2, B_3, B_4, B_5.

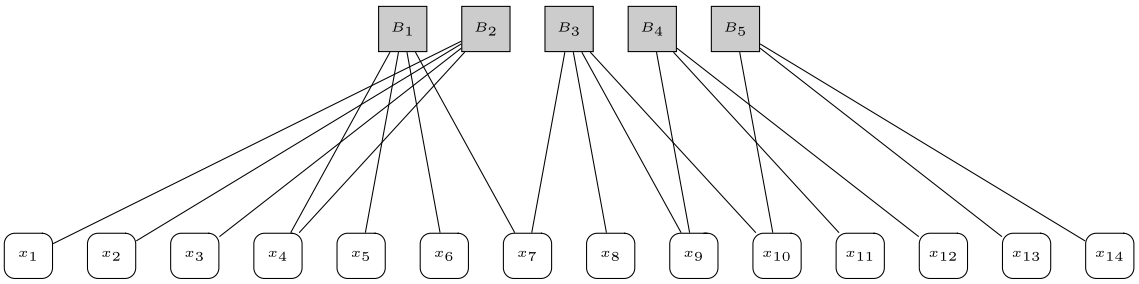


Рисунок 1.13: Графовое представление задачи ДО: гиперграф

Построим двойственный граф для задачи ДО 1.14 из примера (1.8). Множеством вершин двойственного графа 1.14 для задачи будет множество гиперрёбер гиперграфа, причем пара таких вершин соединяется ребром в двойственном графе, если они имеют общие вершины из X. В данном случае множество вершин — B_1, B_2, B_3, B_4, B_5, а множество рёбер — $(B_1, B_2); (B_1, B_3); (B_3, B_4); (B_4, B_5)$.

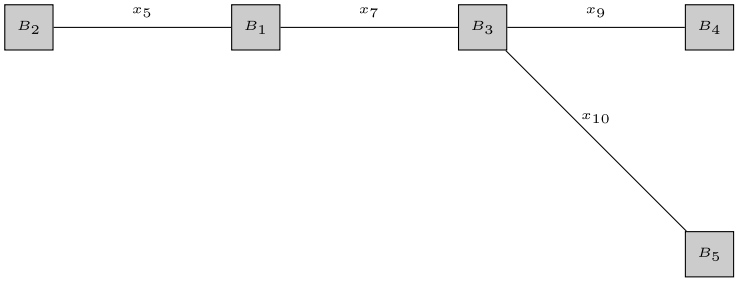


Рисунок 1.14: Графовое представление задачи ДО: двойственный граф

Теперь выделим БЛ–структуру с помощью основного алгоритма. Зафиксируем переменную x_5 и изучим её окрестности возрастающих порядков. Это $S_0 = \{5\}$, $S'_1 = \{4; 6; 7\}$, $S'_2 = \{1; 2; 3; 8; 9; 10\}$ и $S'_3 = \{11; 12; 13; 14\}$. При этом множества индексов ограничений принимают вид: $U'_1 = \{1\}$, $U'_2 = \{2; 3\}$ и $U'_3 = \{4; 5\}$. Определим следующие множества:

$$U_r = \begin{cases} U'_1(x_5) = \{1\}, & r = 1, \\ U'_2(x_5) \backslash U'_1(x_5) = \{2; 3\}, & r = 2, \\ U'_3(x_5) \backslash U'_2(x_5) = \{4; 5\}, & r = 3. \end{cases}$$

Найдём множества, которые содержат сепараторы БЛ–структуры:

$$S_{12} = S(U_1(x_5)) \bigcap S(U_2(x_5)) = \{4; 7\},$$

$$S_{23} = S(U_2(x_5)) \bigcap S(U_3(x_5)) = \{9; 10\}.$$

Далее найдём множества, содержащие индексы столбцов, которые не являются связывающими:

$$S_r = \begin{cases} S\left(U_1\left(x_5\right)\right) \setminus S_{12} = \{5; 6; \}, & r = 1, \\ S\left(U_2\left(x_5\right)\right) \setminus \left(S_{23} \cup S_{34}\right) = \{1; 2; 3; 8\}, & r = 2, \\ S\left(U_3\left(x_5\right)\right) \setminus S_{34} = \{11; 12; 13; 14\}, & r = 3. \end{cases}$$

Таким образом, БЛ–структура состоит из трёх блоков $\{U_1, U_2, U_3\}$, которые выглядят следующим образом: (рис. 1.15).

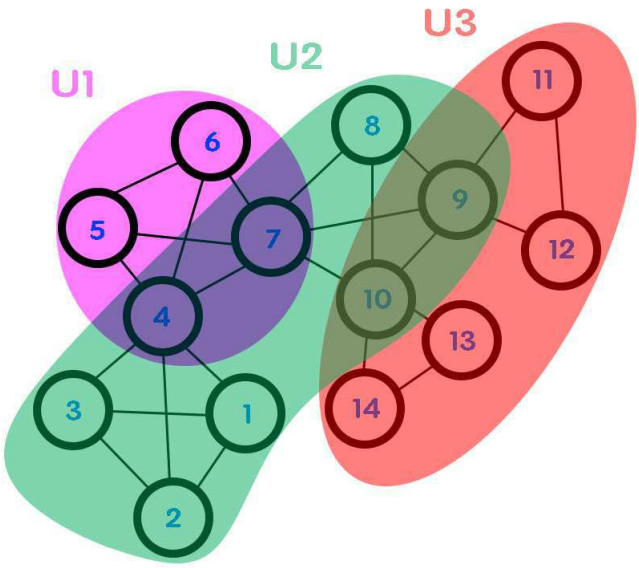
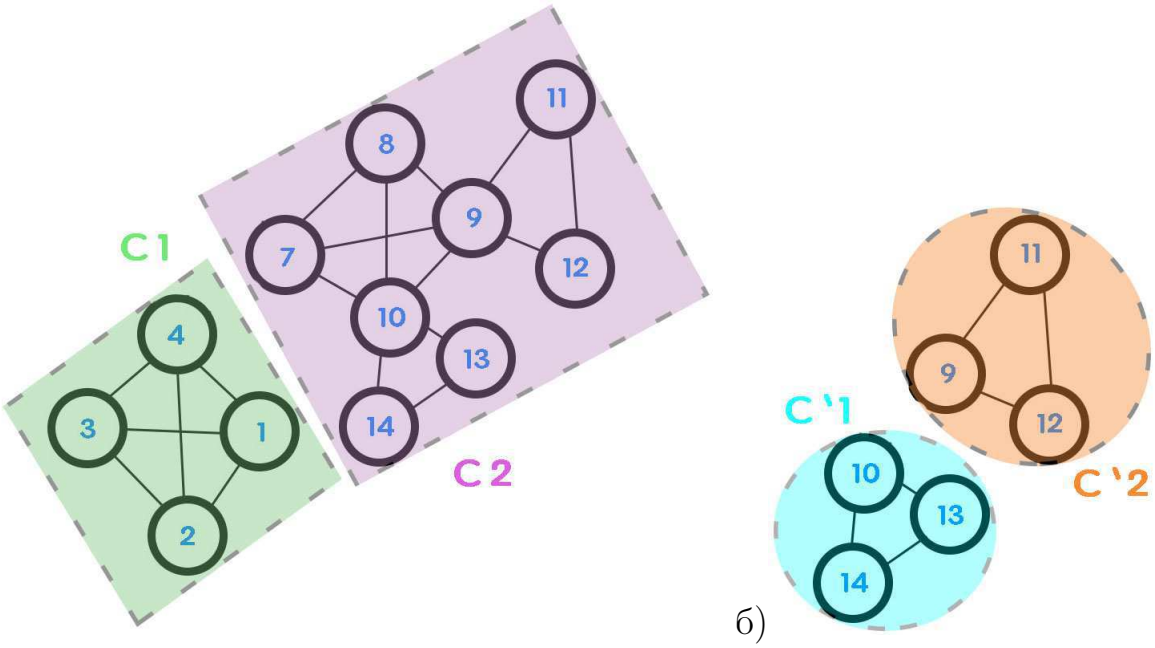


Рисунок 1.15: Выделение БЛ–структуры $\{U_1, U_2, U_3\}$ с помощью основного алгоритма

Далее с помощью процедуры выделения подблоков найдём подблоки в полученной структуре. (рис. 1.16). Рассмотрим сепаратор $S_{12} = \{4; 7\}$. Вершины x_4 и x_7 соответствуют элементам, которые находятся в различных строках, и не существует такой строки, для которой бы существовал связывающий столбец относительно данных элементов. Значит блок U_2 можно расщепить на подблоки соответственно сепаратору S_{12}. Вместо блока U_2 получим подблоки C_1 и C_2, где $C_1 = \{2\}, S_{C_1} = \{1; 2; 3\}$ и $C_2 = \{3\}, S_{C_2} = \{8\}, S_{C_2,3} = \{9; 10\}$. Рассмотрим сепаратор $S_{C_2,3}$. Переменные x_9 и x_{10} находятся в различных ограничениях, и не существует такого ограничения, которое содержало бы обе эти переменные. Значит блок C_2 можно расщепить на подблоки соответственно сепаратору $S_{C_2,3}$. Вместо блока C_2 получим подблоки C_1' и C_2', где $C_1' = \{5\}, S_{C_1'} = \{13; 14\}$ и $C_2 = \{4\}, S_{C_2} = \{11; 12\}$.



а) б)

Рисунок 1.16: Выделение подблоков: а)C_1 и C_2 б)C_1‘ и C_2‘

Таким образом, в результате работы процедуры выделения подблоков получим новую БД–структуру степени $\rho = 3$, которая состоит из пяти блоков $\{U'_1 = U_1, U'_2 = C_1, U'_3 = C_2, U'_4 = C'_1, U'_5 = C'_2\}$, которые выглядят следующим образом:

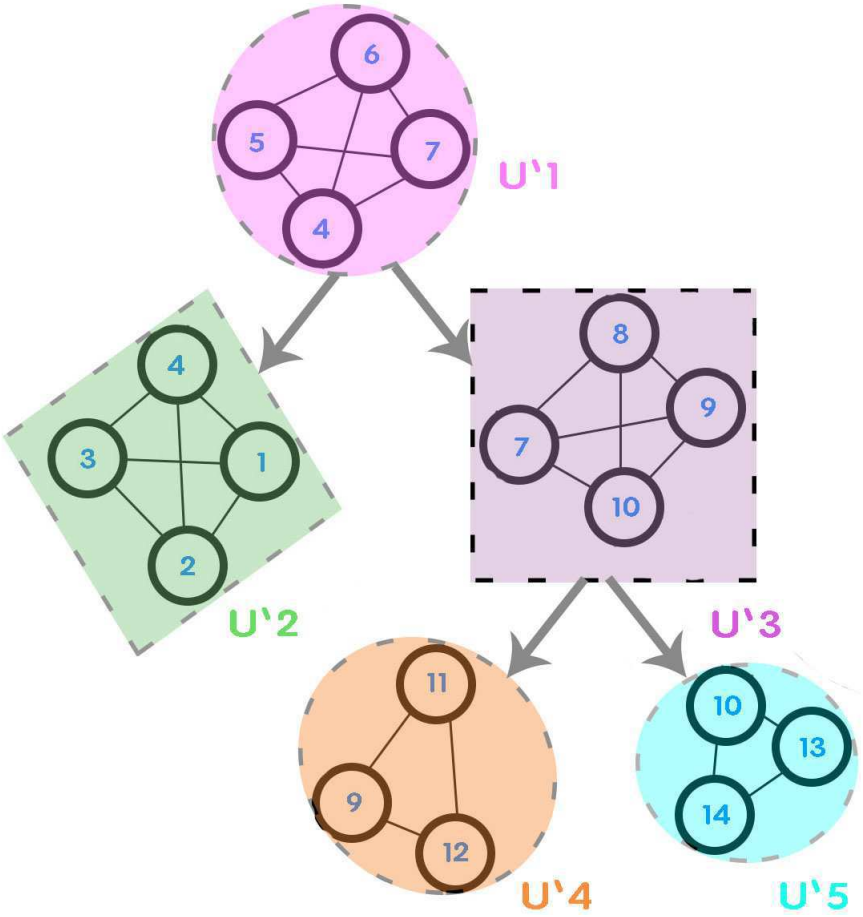


Рисунок 1.17: БД–структура $\{U_1‘, U_2‘, U_3‘, U_4‘, U_5‘\}$

Заметим, что в данной задаче отсутствовали ограничения, которые не содержали сепараторов между блоками. В случае достаточно больших задач ДО имеет смысл рассматривать «промежуточные» сепараторы внутри блоков для выделения дополнительных элементов БД–структуры.

Глава 2

Порядок исключения переменных в локальном элиминационном алгоритме

Локальный элиминационный алгоритм представляет из себя декомпозиционный итерационный метод, где на каждом шаге фиксируется (исключается, элиминируется) переменная или группа переменных. Они принимают фиксированные значения 0 или 1, если речь идёт о булевых постановках. При этом оказывается, что правила выбора элиминации влияет на скорость работы алгоритма. Правила исключения формулируются в терминах понятия теории графов. В данной главе устанавливаются элиминационные правила, а также вводятся понятия и доказываются свойства графовых структур, соответствующих порядку элиминации. Это в частности позволяет утверждать, что задача об оптимальном выборе порядка является NP–полной.

2.1 Методы декомпозиции в целочисленном программировании

Ниже рассматриваются разреженные задачи ДО. Рассмотрим определение из [143].

Определение 2.1. Задачей дискретной оптимизации называется задача следующего вида:

$$Z = extr_{x \in \mathcal{G}} f(x), \tag{2.1}$$

где $\mathcal{G}$ — конечное или счётное множество допустимых решений, а $extr$ — экстремум функции, то есть её минимум или максимум. При этом x называется вектором решений, а $f(x)$ — целевой функцией задачи.

Зададим множество $\mathcal{G}$ системой ограничений: $g_i(x) \gtreqless 0, i \in M$, где $g_i(x)$ — функция для ограничения i, множество индексов ограничений $M = \{1, \ldots, m\}$, множество индексов переменных задачи — $N = \{1, \ldots, n\}$. Если соответствующая задача ДО имеет целочисленные компоненты вектора решений $x = (x_1, \ldots, x_n) : x_j \in \mathbb{Z}, j \in N' \subset N$, то такую задачу будем называть задачей частично–целочисленного программирования. Если $N' = N$, будем называть такую задачу задачей целочисленного программирования. Рассмотрим определение задачи целочисленного программирования из [81].

Определение 2.2. *Задачей целочисленного линейного программирования (ЦЛП) называется задача, которая формулируется следующим образом:*

$$extr\{cx \mid Ax \gtreqless b, \text{ все компоненты } x \text{ — целочисленные}\}. \tag{2.2}$$

Здесь c — вектор целевой функции, A — матрица ограничений, а b — вектор ограничений.

Замечание. Соответственно, задача частично–целочисленного линейного программирования (ЧЦЛП) будет формулироваться так:

$$extr\{cx \mid Ax \gtreqless b, \text{ существуют целочисленные компоненты } x\}. \tag{2.3}$$

Далее будем рассматривать целочисленную задачу линейного программирования с булевыми переменными:

$$extr\{cx \mid Ax \gtreqless b, x_j \in \{0, 1\}, j \in N\}. \tag{2.4}$$

Модели задач с булевыми переменными, т.е. переменными, принимающими только два значения «1», или «0», представляют важнейший класс задач ДО. С помощью булевых переменных можно моделировать и решать задачи, в которых надо выбирать решение из имеющихся различных вариантов. Такие задачи встречаются в различных прикладных областях: задача о назначении, темпоральная задача о ранце, задача коммивояжёра, задача выбора вариантов, задача размещения и т.д.

Рассмотрим задачу целочисленного линейного программирования с булевыми переменными (2.4) на максимум в общем виде:

$$f(X) = \sum_{j=1}^{m} c_j x_j \to \max \tag{2.5}$$

с ограничениями

$$\sum_{j=1}^{m} a_{ij} x_j \le b_i, i = 1, 2, \ldots, n; \tag{2.6}$$

$$x_j \in \{0; 1\}, j = 1, 2, \ldots, m; \tag{2.7}$$

где $x = (x_1, x_2, \ldots, x_m) \in X$ — множество переменных.

Под разреженной задачей ЦЛП будем понимать такую задачу, матрица ограничений которой является разреженной. Построим пример задачи ЦЛП. (2.5)–(2.7).

$$x_1 + x_2 + 4x_3 + 3x_4 + 2x_5 + 5x_6 + 2x_7 \to \max \tag{2.8}$$

$$\begin{cases} x_2 + 2x_3 + 4x_4 \le 6 \\ 3x_1 + 2x_2 + 2x_3 \le 4 \\ 8x_2 + 4x_5 \le 8 \\ x_3 + 3x_6 + 3x_7 \le 4 \\ x_j \in \{0, 1\}, j = 1, 2, \ldots, 7. \end{cases} \tag{2.9}$$

Подразумевается, что вершины графа взаимосвязей матрицы A и вершины графа ограничений соответствующей задачи ЦЛП эквивалентны.

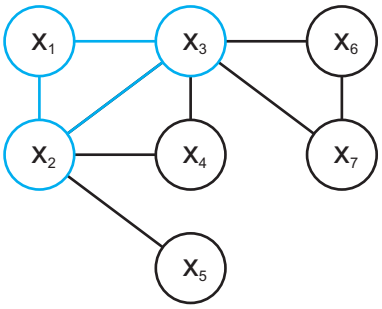


Рисунок 2.1: Граф ограничений, описывающий ограничения задачи ДО из примера (2.8)

В [144] рассматриваются задачи частично–целочисленного линейного программирования (ЧЦЛП) с блочной структурой. Здесь под блочной структурой понимается задача вида (2.10–2.12). Целевая функция задачи задаётся следующим образом:

$$\sum_{i=1}^{p}(C_i x_i + {C'}_i y_i) + C'' z \to \min; \tag{2.10}$$

Ограничения задач ЧЦЛП с блочной структурой делятся на два типа: связывающие и блочные. Связывающие ограничения задачи имеют вид:

$$\sum_{i=1}^{p}(A_{ij} x_i + {A'}_{ij} y_i) + A'' z = b_{0j},\ j \in [1 : m_0]. \tag{2.11}$$

Блочные ограничения задачи имеют вид:

$$B_i x_i, + {B'}_i y_i + B'' z = b_i, i \in [1 : p]. \tag{2.12}$$

Рассмотрим компоненты целевой функции и ограничений. В задаче вида (2.10–2.12) участвуют два вектора решений задачи — дискретный вектор решений $x_i \in \mathbb{Z}_+^{n_i}$ и непрерывный вектор решений $y_i \in \mathbb{R}_+^{n'_i}$, где $i \in [1 : p]$, а также связывающая переменная $z \in \mathbb{R}_+^{n''}$, которая может быть непрерывной или целочисленной в зависимости от конкретной модели задачи. В общем случае мы будем понимать под связывающей переменной z вектор связывающих переменных $\mathbf{z}_k$, где $\mathbf{z}$ — блок переменных типа z. Дискретному вектору решений x_i соответствуют вектор целевой функции C_i размерности n_i, матрица ограничений задачи A_{ij} и B_{ij} — матрицы размерностей $M_0 \times N_i$ и $m_i \times n_i$ соответственно, $i \in [1 : p], j \in [1 : m_0]$. Непрерывному вектору решений y_i соответствуют вектор целевой функции ${C'}_i$ размерности ${n'}_i$, матрица ограничений задачи ${A'}_{ij}$ и ${B'}_{ij}$ — матрицы размерностей $m_0 \times {n'}_i$ и $M_i \times {N'}_i$ соответственно, $i \in [1 : p], j \in [1 : m_0]$. Переменной z соответствуют компонента целевой функции C'', вектор ограничений задачи ${A''}_j$ размерности m_0 и ${B''}_{ij}$ — матрица размерности $M_i \times N''$, $i \in [1 : p], j \in [1 : m_0]$. При этом $b_{0j}, j \in [1 : m_0]$ — компоненты вектора ограничений, соответствующие связывающим ограничениям, а $b_i, i \in [1 : p]$ — векторы размерности m_i, соответствующие блочным ограничениям.

Обратим внимание, что в данной работе будут рассматриваться связывающие ограничения вида $\sum_{i=1}^{p} A_{ij} x_i + A''_{ij} z_j = b_{0j},\ j \in [1 : m_0]$, где x_i — свободные

переменные, которые входят только в ограничения своего блока, а z_j — целочисленные связывающие переменные, которые входят в ограничения нескольких блоков. A_{ij}, A''_{ij} и b_{0j} — соответствующие коэффициенты при переменных, , $i \in [1:p], j \in [1:m_0]$. В дальнейшем мы будем разделять x_i и z_j согласно множествам их индексов.

Далее в [144] приводится классификация блочных задач ЧЦЛП. По типу связей между подсистемами блочные задачи ЧЦЛП делятся следующим образом:

— если $n'' = 0$, то задача не содержит связывающих переменных, иначе — содержит;

— если $m_0 = 0$, то задача не содержит связывающих ограничений, иначе — содержит.

Заметим, что в данной работе рассматриваются задачи дискретной оптимизации, которые содержат связывающие переменные, но не содержат связывающих ограничений.

Далее рассмотрим блочные задачи ЧЦЛП по наличию непрерывных переменных в блоках:

— если $n'' = 0, n'_i = 0, i \in [1:p]$, то задача является задачей ЦЛП;

— в противном случае задача является задачей ЧЦЛП.

Рассмотрим классификацию по типу связывающих ограничений: связывающие ограничения (2.11) могут содержать ненулевые коэффициенты только при целочисленных или непрерывных переменных, либо содержать коэффициенты только одного знака, либо обладать какой–либо спецификой, которая будет использована при построении алгоритма для решения задачи ДО. Классификация по типу подсистем подразумевает, что структура блочных ограничений существенно влияет на эффективность применения итерационных методов декомпозиции. В данной работе связывающие ограничения не используются.

В [144] рассматриваются частные случаи блочных задач ЧЦЛП: задача о рюкзаке, задача транспортного типа и её модификации, задачи о разбиении и покрытии, а также распределительная задача с булевыми переменными. Также рассматриваются блочные модели отраслевого планирования, в системах обработки данных и некоторые другие блочные целочисленные задачи ДО.

Так как блочные задачи встречаются на практике достаточно часто, разработано множество алгоритмов решения оптимизационных задач с блочной структурой, которые называются алгоритмами блочного программирования. Алгоритмы обработки разреженных задач направлены на извлечение определённых структур, то есть получение нулей на нужных позициях матрицы. При этом возникают новые ненулевые элементы в ещё необработанной части матрицы. Р. Тьюарсон в [145] решает задачу минимизации этого процесса, поскольку весь процесс решения можно промоделировать заранее и выбрать оптимальный порядок шагов алгоритма. Эта же идея лежит в основе выделения блочных структур в задачах ДО. В статье [146] анализируется возможность использования программного пакета FBDK (Function Block Development Kit) в качестве инструмента для разработки и реализации задач оптимизации управления технологическими процессами. Оценка инструмента осуществляется с точки зрения начинающего пользователя путем разработки алгоритма Particle Swarm Optimization (PSO), который используется из–за его легкого внедрения и надежности. В статье [147] блочное программирование используется для разработки и оценки трансмиссии трактора. В [148] рассматривается робастная схема выделения блочных структур и исследуется эффективность такого алгоритма. В [149] исследуется выделение блочной структуры с помощью программного пакета SCOTCH, причём рассматриваются задачи с линейными ограничениями, при этом целевая функция является квадратичной.

Перейдём к выделению блочной структуры задач, описанных выше, в контексте индексов переменных. Из множества индексов вектора решений N выделим подмножества $N_1, \ldots, N_m$, которые соответствуют ограничениям $g_i(x)$. Объединим подмножества $N_1, \ldots, N_m$ в группу множеств $\tilde{N}_1, \ldots, \tilde{N}_{\tilde{m}}$, $m \geq \tilde{m}$ таким образом, чтобы каждое N_i входило в одно и только в одно множество $\tilde{N}_r$, $r \in \{1, \ldots, \tilde{m}\}$. Рассмотрим полученную блочную структуру $B_1, \ldots, B_{\tilde{m}}$, где B_r — блок, то есть подмножество переменных $x_j \in B_r$, номера которых $j \in \tilde{N}_r$. Если множества $B_1, \ldots, B_r$ можно упорядочить таким образом, что у каждого блока будут общие элементы с предыдущим и последующим множеством и не будет общих элементов ни с каким другим множеством, будем называть такую задачу ЦЛП задачей с БЛ–структурой.

Класс БЛ–задач линейного программирования — задач для динамического планирования, был среди первых классов разреженных задач линейного про-

граммирования большой размерности (см. Dantzig [150]). James Ho и Etienne Loute опубликовали следующие примеры БЛ–задач ЛП: задачи составления назначений, задачи для многоэтапного планирования, задачи расписаний и задачи многоэтапного структурного проектирования [151]. Темпоральные задачи о ранце [152], которые достаточно недавно начали применяться для решения задач предварительного резервирования вычислительных ресурсов в Grid Computing, некоторые задачи управления трудовыми ресурсами [153], а также управления в иерархических (как правило, в древовидных) структурах, задачи линейного динамического программирования [154], динамические модели экономики, которые учитывают фактор времени в явном виде [155], задачи многоэтапного стохастического программирования, сетевые задачи также имеют БЛ–структуру. Под структурной декомпозицией задачи будем понимать выделение и упорядочивание её блочной структуры для дальнейшего использования. Ниже приведём краткий обзор современных методов декомпозиции задач ЦЛП и ЧЦЛП.

В [3] изучаются методы декомпозиции для блочных задач целочисленного программирования. Там представлен обзор по данной проблематике до середины восьмидесятых годов прошлого столетия. Ниже предлагается краткий обзор последующего периода. Разумеется, он не претендует на полноту, далее представлены некоторые оригинальные подходы.

В [156] рассматривается использование модели задачи смешанного целочисленного программирования для решения задач управления планированием использования рабочей силы и её возможной транспортировки. В этой задаче множество исполнителей назначается на выполнение заданий, места выполнения которых распределены географически. Такая постановка возникает в сценариях планирования работы социальных работников в Великобритании. Авторы представляют задачу смешанного целочисленного программирования, которая отражает важнейшие особенности практической задачи, такие как наличие определенных географических районов и приспособляемость исполнителей (их доступность и различную мобильность). В статье показано, что на качество итогового решения влияет порядок, в котором решаются подзадачи. Поэтому авторы исследуют различные способы упорядочивания подзадач, и показывают, что весьма перспективным является подход на основе декомпозиции, который

предоставит решения высокого качества за разумное время и используя точный метод оптимизации.

В статье [157] было обнаружено, что в задачах назначения и планирования, ПО для решения задач на основе смешанного целочисленного программирования стало конкурировать или превосходить подходы, основанные на методе Бендерса (использующего логическую декомпозицию задачи). Приведенные в той статье реализации метода декомпозиции Бендерса, тем не менее, отличаются от описанных в цитированной авторами литературе. Это приводит к тому, что производительность решения оказывается существенно ниже, чем указанно в источниках. В [158] обнаружили, что для задач большей размерности, при корректной реализации метода Бендерса, он остаётся на 2–3 порядка быстрее, чем самые новейшие коммерческие системы решения задач на основе подходов смешанного целочисленного программирования, что полностью делает противоположными выводы предыдущей статьи.

В работе [159] предлагается архитектура для решения задач построения оптимальных траекторий, декомпозиция в которой устроена следующим образом: глобальная задача обхода препятствий разбивается на простые подзадачи, соответствующие гомотопиям различных путей. В классических подходах к планированию траекторий на основе гомотопий, планирование траектории и идентификация гомотопий проводятся одновременно, что приводит к большой вычислительной сложности. В данной статье предлагается метод, позволяющий перечислять и явно выражать различные классы гомотопий до шага планирования траектории или оптимизации, что позволяет расщепить задачу на более простые независимые подзадачи. В статье изложено два интересных результата. Первый результат - это описание метода, который использует известные методы н ячеевидной декомпозиции (cell decomposition), чтобы перечислить и описать все созданные локальные задачи, которые могут быть решены эффективно и независимо. Кроме того, проанализировано отношение между предложенными представлениями ячейка-последовательность (cell-sequence representation), и классами гомотопий. Второй результат представляет собой новый, вычислительно эффективный метод решения задачи оптимизации траекторий в виде последовательности ячеек, основанный на смешанном целочисленном квадратичном программировании. При помощи симуляции показаны вычислительная эффективность и большое множество решений. Предложенная формулировка

задачи смешанного целочисленного квадратичного программирования хорошо вписывается в класс подходов на основе предсказаний для линейных моделей с невыпуклыми ограничениями на отсутствие столкновений.

В [160] авторы рассматривают класс двухэтапных задач стохастического целочисленного программирования, где на первом этапе переменные являются бинарными, а на втором - целочисленными общего вида. Разработан алгоритм декомпозиции, аналогичный методам L-разбиения и Бендерса, где при помощи сечения Гомори, итеративно получаются всё более и более точные приближения решений целочисленных задач второго этапа. Авторы показывают, что предложенная методология является гибкой, допуская различные способы реализации, каждый из которых даст алгоритм, сходящийся за конечное число шагов. Авторы описывают свои алгоритмы, используя примеры из литературы. Также в статье приводятся вычислительные результаты для нескольких случаев стохастической задачи местонахождения сервера (server location problem), которые показывают что указанный алгоритм, основанный на декомпозиции, лучше масштабируется по количеству сценариев, чем другой известный современный алгоритм решения/решатель, использованный для решения эквивалентной детерминированной задачи.

В работе [161] предложен новый метод отсекающих плосткостей для двухэтапных задач стохастического смешанного целочисленного программирования, названный декомпозицией Фенхеля (Fenchel decomposition, FD). FD (декомпозиция Фенхеля) использует класс выполняющихся неравенств, называемый отсечения Фенхеля, которые получаются при помощи отсекающих плоскоскостей Фенхеля в задачах целочисленного программирования. В начале, авторы получают отсечения Фенхеля, использующие переменные как первого, так и второго этапов, формулируют алгоритм на основе отсечений Фенхеля для задач стохастического смешанного целочисленного программирования и показывают сходимость за конечное время для случая бинарного первого этапа. Также в статье получены отсечения Фенхеля, содержащие только переменные второго этапа, и использованы идеи из дизъюктивного программирования (disjunctive programming) для расширения отсечений в пространстве большей размерности, включающее в себя переменные первого этапа. После этого приводится альтернативный алгоритм (FD-L), основанный на расширенных отсечениях. В завершение приводятся результаты вычислительных экспериментов для нескольких

тестовых задачах, известных из литературы, имеющих специальную структуру задачи о рюкзаке с неотрицательными коэффициентами в левой части. Результаты выглядят перспективно, оба алгоритма в случае больших размерностей превосходят прямой алгоритм решения, и прямой алгоритм дизъюктивного программирования (disjunctive decomposition). Более того, алгоритм FD–L в целом показывает лучшие результаты, чем алгоритм FD. Поскольку отсечения Фенхеля в общем случае могут быть вычислительно сложными и лучше всего приспособлены для задач со специальной структурой, оба алгоритма существенно используют специальную структуру тестовых задач, уменьшая размерность задач генерации отсечений, основываясь на количестве ненулевых компонент в нецелочисленном решении, которые необходимо отсечь.

В [162] говорится, что объединение задач планирования выпуска продукции и динамической оптимизации может увеличить общую производительность многопродуктовых реакторов непрерывного действия с механическим перемешиванием. Однако, такое объединение приводит к появлению смешанной целочисленной задачи динамической оптимизации большой размерности, которая может оказаться очень сложной в решении. Авторы предлагают два эффективных метода решения на основе обобщенного подхода декомпозиции Бендерса, который использует специальную структуру объединенной задачи. Первый метод использует подход с разделением по времени согласно парадигме “Master–workers”. После декомпозиции главная задача представляет собой набор отделившихся задач динамической оптимизации (worker problems) и координирующую задачу (master problem), которая является задачей смешанного целочисленного нелинейного дробного программирования. Затем координирующая задача решается до нахождения глобально оптимального решения методами дробного программирования, что гарантирует допустимость отсечений Бендерса. Второй метод декомпозиции применяется к задаче вычисления оптимальной последовательности производства. Аналогично первому методу, во втором используется алгоритм дробного программирования для решения координирующей задачи. По сравнению с одновременным методом, в задаче оптимизации производственного процесса в многопродуктовых реакторах непрерывного действия с механическим перемешиванием, предложенные алгоритмы декомпозиции могут уменьшить время вычисления на два и более порядка.

В работе [163] рассматриваются диффузионные процессы в производстве полупроводников, представляющие собой последовательность процессов от очистки пластины до процесса отжига. Большинство печей производит обработку поступающих пластин крупными партиями за одну процедуру, и этот процесс характеризуется сравнительно большими затратами по сравнению с другими технологическими процессами полупроводниковых производств. Строгое ограничение по времени связывает между собой процесс очистки пластины и процессы отжига для дальнейшего контроля качества. Эти производственные ограничения на диффузионные процессы делают задачи составления расписаний очень сложными. В статье предлагается улучшенный подход к задачам составления расписаний на основе понятия скользящего горизонта планирования. В силу комбинаторной природы задачи планирования сложность задачи экспоненциально растет при увеличении количества заданий и инструментов. Однако, в практических случаях время вычислений должно быть ограничено, поскольку в большинстве полупроводниковых производств требуется, чтобы расписание обновлялось за короткий интервал времени. Авторы предлагают модель оптимизации диффузионных процессов на основе задачи смешанного целочисленного линейного программирования, а также эффективный метод декомпозиции этой сложной задачи. Предлагаемый метод декомпозиции, повторяет процесс составления расписания по мере того, как он постепенно увеличивает количество запусков, что позволяет алгоритму планирования создавать расписания, близкие к оптимальным на ограниченных интервалах времени. Алгоритм планирования может существенно увеличить ключевые показатели производительности, такие как частоты нарушений ограничений по времени, размеры партий, пропускную способность. В статье также рассматривается программная архитектура для реализации алгоритма планирования.

В [164] авторы предлагают новый подход для точного решения задач математического программирования с вероятностными ограничениями (chance-constrained mathematical programs), которые имеют дискретные распределения с конечным носителем и случайными ограничениями типа многогранника. Такие задачи известны своей чрезвычайной сложностью за счёт невыпуклости области допустимых решений. Большинство известных методов способны всего лишь найти доказуемо хорошие решения в некоторых весьма узких частных случаях. Предлагаемый подход использует декомпозицию для получения под-

задач,каждая из которых соответствует одному возможному исходу, и методы целочисленного программирования для объединения результатов решения подзадач, а затем получения детерминированных сильных выполняющихся неравенств. Вычислительные эксперименты для задачи математического программирования с вероятностными ограничениями относительно задачи планирования ресурсов в задаче оптимизации персонала контактного центра показывают, что предложенный подход работает существенно лучше, чем существующие постановки на основе смешанного целочисленного программирования, а также чем простой декомпозиционный метод, который не использует детерминированных неравенств. Авторы также показывают, как данный подход может применяться для эффективного нахождения последовательности уровней риска в приложении к задаче нахождения эффективной зависимости между риском и стоимостью.

В [165] предлагается метод объединения алгоритмов прогрессивного хеджирования (Progressive Hedging, PH) и двойственной декомпозиции (Dual Decomposition, DD, Каро и Шульц) для задач стохастического смешанного целочисленного программирования. Используя соответствие между нижними оценками, полученными в алгоритмах PH и DD, найден метод преобразования весов из PH в множители Лагранжа. Быстрый прогресс на начальных итерациях алгоритма PH ускоряет сходимость алгоритма DD к точному решению. В статье приводятся результаты вычислительных экспериментов для задач о положении сервера или единичных обязательствах (unit commitment)

В [166] описывается алгоритм декомпозиции, который сочетает декомпозицию Бендерса и основанную на сценарии лагранжеву декомпозицию, в применении к двухэтапной задаче стохастического программирования о планировании инвестиции с полным регрессом. Переменные первого этапа являются смешанно–целочисленными, а второго — непрерывными. Алгоритм основан на схеме кросс–декомпозиции (перекрестной декомпозиции), и полностью совместно использует информацию из прямой и двойственной задач, в части добавленных прямодвойственных мульти–отсечений (разрезов) в координирующие лагранжеву задачу и задачу Бендерса, для всех сценариев. Преимущества схемы перекрестной декомпозиции показаны на наглядном примере задачи о расположении производства в условиях риска сбоев.

В работе [167] представлен гибридный метод решения задачи планирования развития радиальных сетей распространения с распределенными генераторами. Этот метод сочетает в себе подход на основе метода отжига и задачу смешанного линейного целочисленного программирования. Задача планирования расширения сначала рассматривается как оптимизационная задача смешанного линейного целочисленного программирования с целевой функцией, минимизирующей стоимость инвестиций, величины потерь, стоимость прерывания обслуживания потребителей по причинам сбоев в производственных подразделениях или в распределенных генераторах, и стоимость потерянных распределенных генераторов, связанную со сбоями в производственных подразделениях. Чтобы уменьшить сложность задач планирования, предлагается декомпозиция задачи на семейства последовательностей подзадач (локальных сетей), которые решаются с использованием модели смешанного линейного целочисленного программирования. Указанная декомпозиция и сам процесс поиска оптимального решения находятся под управлением и итеративным уточнением от предложенного алгоритма отжига, который использует соответствующий механизм интенсификации и диверсификации для получения минимальной полной стоимости решения.

В работе [168] предлагается новый метод помогающий оперативно управлять действиями по планированию в реальных трубопроводах, перекачивающих тяжелые нефтепродукты, которые имеют меньшую суммарную стоимость, такие как, например, мазут или судовое топливо. Эти нефтепродукты обладают особыми характеристиками, которые существенно влияют на процесс их транспортировки, по причине невозможности перекачки при комнатных температурах, большой вязкости или использования общих резервуаров для нескольких видов продуктов. Таким образом, во время транспортировки нефтепродуктов такого типа вся цепочка трубопроводов и резервуары должны поддерживаться в нагретом состоянии на протяжении всего процесса перекачки. Такие требования приводят к необходимости разработки специальной модели, ориентированной на такой класс задач. Предложенный в работе метод решения основан на процедуре декомпозиции, которая использует последовательность задач математического программирования, с применением некоторых эвристик. Приведенный подход проверен на реальном сценарии с сетью трубопроводов древовидной структуры.

В [169] авторы исследуют задачу планирования сети перевозок морскими судами. Эта задача заключается в том, чтобы для заданного флота грузовых кораблей, которые в совокупности перевозят различные типы товаров, получить множество циклических маршрутов, не являющихся простыми циклами. Цель оптимизационной задачи состоит в максимизации прибыли грузового транспорта при минимизации операционных затрат. Существенный потенциал методов исследования операция для проектирования морских сетей транспортировки, эффективных по стоимости и энергозатратам, остается нераскрытым в литературе. Известны эффективные средства планирования логистики перевозок для авиалиний, железнодорожного транспорта, для компаний других различных видов перевозок, но в области морской транспортировки использование методов исследования операций для оптимизации логистики имеет малый масштаб применения. Возможно, отсутствие знания предметной области и малое количество реальных данных являются барьером для исследователей, препятствующим разработке эффективных сетей морских перевозок. В данной статье разработан набор тестовых задач, что помогает облегчить доступ специалистов по исследованию операций к данным из рассматриваемой предметной области морских перевозок. Описывается и приводится анализ предметной области морских перевозок в контексте задачи проектирования оптимальной сети перевозок. Также приводится разносторонняя модель целочисленного линейного программирования на основе сервисов, создающих фиксированные маршруты для морской транспортной компании. Доказывается, что задача разработки оптимальной сети морских перевозок является сильно NP–трудной. Приводится набор тестовых задач, основанных на данных, отражающих реальную структуру международной транспортной сети. Принципы разработки этого набора данных рассматриваются в связке с промышленными стандартами, деловыми обычаями и моделями математического программирования. В основе набора данных лежат реальные данные от крупнейшей международной морской транспортной компании MaerskLine, дополненные другой информацией от некоторых бизнесменов в данной области. Приведены результаты расчетов, демонстрирующие наилучшие на данный момент решения для шести из семи задач набора, используя эвристические комбинации метода поиска с запретами и эвристический метод генерации столбцов.

В работе [170] предложен эффективный подход на основе метода Бендерса, для решения задачи о вводе в эксплуатацию источников переменного электрического тока с ограничениями на уровне сети, в условиях неопределенности. Единственным источником неопределенности в этой работе принимается производство электроэнергии ветряными электростанциями, что моделируется при помощи соответствующего набора сценариев. Предложенная модель имеет вид двухэтапной задачи стохастического программирования, где первый этап относится к рынку электроэнергии с расчетами «завтра», а второй этап представляет производство в реальном масштабе времени. При помощи метода Бендерса предлагается декомпозиция исходной в общем случае трудно разрешимой задачи, относящейся к классу смешанного целочисленного нелинейного программирования, на комбинацию из главной задачи типа смешанного линейного целочисленного программирования и набора нелинейных, но непрерывных задач, по одной на каждый сценарий. Кроме того, чтобы провести декомпозицию задачи о вводе в эксплуатацию источников переменного тока, предложен эвристический подход с релаксацией ограничений по времени для генерирующих источников. Полезность предлагаемого подхода показана на примере численного решения типовой задачи IEEE о системе тестирования надежности для одной области.

В статье [171] рассматривается задача частичной ориентированной взвешенной неполной раскраски (Partial Directed Weigthed Improper Coloring Problem, (θ, k)–PDWICP) для заданного вещественного θ и целого k, которая заключается в вычислении размера наибольшего подграфа G' в G, такого что G' допускает θ–неполную k–раскраску. Задача (θ, k)–PDWICP является естественной моделью при решении задачи присвоения каналов с целью максимизации количества одновременно обслуживаемых мобильных терминалов в беспроводной сети. В работе авторы сравнивают подходы на основе целочисленного программирования для точного решения этой NP–трудной задачи. Применяется метод ветвей и границ с использованием полиномиально вычислимой оценки сверху, получаемой по аналогии с $\theta(G)$ функцией Ловаса, и использованием ограничений из задач о рюкзаке и упаковке. Приведено сравнение с методом ветвлений и оценок.

В статье [172] приводится модификация метода Бендерса LBBD, которая задаёт новый уровень качества решения различных задач планирования и со-

ставления расписаний, в том числе благодаря взаимодополняющему сочетанию сильных сторон методов программирования в ограничениях и смешанного целочисленного программирования. Приводится вычислительный анализ конкретных факторов, которые обуславливают успех LBBD, выработаны рекомендации для будущих реализаций. Изучается класс задач, в котором задания распределяются по многим ресурсам и ставится общая задача планирования на каждом из ресурсов. Авторы показывают, что для больших задач метод LBDD как минимум в 1000 раз быстрее, чем самые современные подходы на основе смешанного целочисленного программирования, несмотря на недавние успехи в этой области. Более того, установлено, что метод LBDD наиболее эффективен, когда аспекты задачи связанные с планированием и составлением расписания примерно сбалансированы по сложности. Наиболее эффективный прием, позволяющий улучшить метод LBDD — это включение релаксации подзадач в главную задачу. Также оказалось, что важную роль играет усиление отсечений Бендерса, когда сложности главной задачи и подзадач являются сбалансированными. Эти результаты дают почву для направления дальнейших исследований.

В работе [173] изучается задачи планирования нагрузки на серверы, в системах с несколькими классами обслуживания, при неопределенности в количестве прибывающих пользователей. В таких системах обычно сначала определяют величину исполнительных ресурсов, а затем определяют расписания загрузки исполнителей, которые покрывают имеющиеся ресурсы. Предлагается новая модель на основе стохастического целочисленного программирования, которая объединяет эти два шага, что может дать меньшие затраты за счет возможного использования альтернативных конфигураций серверов, которые могли бы давать сравнимый уровень обслуживания. Авторы показывают, что метод ветвей и границ, основанный на декомпозиции Бендерса может не доставлять правильного решения при слабых оценках границы релаксации. Они предлагают новый подход, основанный на применении округления в смешанно–целочисленном случае, чтобы улучшить отсечения Бендерса используемые в рассматриваемом алгоритме. Такой подход применим к любым задачам стохастического целочисленного программирования с целочисленными переменными первого этапа. Численные эксперименты показывают вычислительную эффективность предложенного подхода и потенциальную преимущества при решении

интегрированных моделей, по сравнению с отдельным рассмотрением задач об оснащении и о составлении расписания.

Разработка и эксплуатация энергетических систем являются ключевыми факторами для согласования спроса и предложения энергетических ресурсов. В работе [174] представлена систематическая процедура, включающая процесс разработки, методы накопления энергии для задач определения размера и оптимизации производства мульти–генерирующих технологий. Элементом новизны является агрегация ресурсов биомассы, а также одновременная многокритериальная и многопериодная оптимизация. Одновременное рассмотрение всех вышеперечисленных аспектов делает задачу трудно решаемой. Для решения такой сложной задачи предлагается подход на основе декомпозиции. В этой статье также предлагаются варианты агрегирования биомассы в энергетическую систему, такие как паровые турбины с противодавлением, циклы Ренкина для биомассы, газовые двигатели и газотурбинные установки с газификацией от биомассы, производство синтетического природного газа, и комбинированные циклы газификации биомассы. Целью задачи является одновременная минимизация стоимости, а также уровня выбросов угарного газа в атмосферу, для чего используются постановка с многокритериальными эволюционными алгоритмами, и задачи смешанного линейного целочисленного программирования. Предложенная модель продемонстрирована на практическом примере. Результаты показывают, что одновременного производство электрической и тепловой энергии из биомассы и природного газа является устойчивым при заданных предположениях. Более того, в случае постепенного увеличения использования возобновляемых источников энергии с уменьшением использования природного газа, достигается высокая экономия энергии и уменьшение выбросов угарного газа, до 40%. С другой стороны, большая экономическая эффективность 52%, достигается для технологий на основе природного газа.

В различных областях производства имеются законодательные ограничения, приводящие к тому, что принятие решений должно сопровождаться учетом экологических факторов. Автомобильная промышленность во многих странах учитывает экологические факторы, особенно, в странах–членах ЕС, где имеется порядок утилизации старых автомобилей. В Турции, процесс утилизации автомобилей подчиняется директиве об управлении утилизацией старой автотехники, которая официально принята Министерством окружающей среды и леса

Турции в 2009 году. Производители автомобилей обязаны обеспечить бесплатные для потребителя сбор, переработку и утилизацию списанной автотехники. В статье [175] рассматривается задача оптимальной переработки списанных автомобилей, предлагается подход на основе смешанного линейного целочисленного программирования для проектирования сети переработки, включающей различных участников, принимающих участие в этом процессе. Предложенная схема проверяется на реальном примере города Анкара, столицы Турции, второго по величине города в стране. Также предложена модель прогнозирования уровня автомобилизации и числа списываемых автомобилей на основе усредненных долгосрочных трендов в числе списываемых автомобилей. Предложенный пример применения метода и прогноз по численности позволяет лучше понять, как будет работать логистическая сеть с течением времени. Полученные результаты свидетельствуют от том, что число предприятий по переработке и стоимость системы будет увеличиваться, как и число утилизируемых автомобилей.

В работе [176] предлагается алгоритм точного решения двухуровневой задачи смешанного линейного целочисленного программирования при наличии трех упрощающих предположений. Хотя история изучения указанного типа задач продолжается уже несколько десятилетий, и они широко применятся к практическим различным задачам из реальной жизни, известно лишь небольшое число методов их решения. По сравнению с существующими методами, предложенный алгоритм требует более слабых предположений и меньшего их числа, а также неявно рассматривает ограниченные оптимальные, недопустимые, и неограниченные случаи и решает задачу корректно за конечное время. Приводятся результаты вычислительных экспериментов на небольшой коллекции задач двухуровневого смешанного линейного целочисленного программирования, которые были созданы авторами, и свободно доступны в сети Интернет.

В работе [177] рассматривается расширение задачи об укладке контейнеров различного типа по различным отсекам морского контейнеровоза, известной как задача генерального плана отсека (masterbay plan problem, MBPP), в случае большого числа портов, когда в различных портах требуется погрузить или выгрузить различные контейнеры. Последовательность чередующихся операций разгрузки–погрузки определяет эффективность плана укладки. В работе рассматриваются две точные модели смешанного целочисленного программи-

рования, для многопортовой задачи, ориентированные на практические и операционные аспекты реальной задачи. Также рассматриваются вычислительно эффективные постановки для задач смешанного целочисленного программирования с релаксацией, приводятся результаты вычислительных экспериментов из практических приложений. Результаты показывают эффективность предложенных методов и моделей.

Существенная доля грузов в мировой торговле перевозится морским способом в контейнерах. Количество перевозок растет из года в года, и имеет место тенденция к увеличению размера вновь создаваемых кораблей. По этой причине, всё более и более важное значение получает задача об укладке контейнеров, которая является фундаментальной задачей при оптимизации морских перевозок, включает в себя вопрос об оптимальном расположении транспортных контейнеров различных типов по отсекам контейнеровоза, в каждом из портов маршрута судна, чтобы максимизировать эффективность процедур разгрузки и погрузки, а также минимизировать стоимость доставки. Несмотря на такую актуальность, до сих пор задача решается специфическими методами, основанными на имеющемся опыте. В работе [178] предлагается многоэтапная эвристическая процедура декомпозиции, которая учитывает многие сложные аспекты этого класса задач в реальном мире. Предложенный подход имеет хорошую производительность на задачах из реальной практики, и в настоящее время готовится ко внедрению в сектор коммерческого программного обеспечения.

Задача управления потоком транспорта в реальном времени уже более 50 лет остается предметом активных исследований. В последние годы, однако, совместное использование повсеместных датчиков и бесшовных технологий взаимодействия устройств породило запрос на разработку более эффективных методов управления, способных работать в реальной обстановке в режиме реального времени. В статье [179] предлагается быстрый метод декомпозиции для задач сетевой оптимизации с приложением к задаче управления потоком в реальном времени. Данный подход основан на постановке задачи управления сетью в виде задачи нелинейного программирования. Он заключается в использовании метода чередующихся направлений с использованием прямую численную симуляцию в одной из оптимизационных подзадач. Метод хорошо масштабируется до размерностей, соответствующих реальному потоку транспорта в больших

городах. В работе демонстрируется хорошая производительность подхода на синтетических и реальных данных о транспортной сети.

В статье [180] предлагается метод для вычисления оценок снизу в алгоритме прогрессивного хеджирования (PHA) для задач двухуровневого и многоуровневого стохастического смешанного целочисленного программирования. Вычисление таких оценок снизу позволяет оценивать качество решения для указанного алгоритма по мере процесса решения. Оценки снизу можно вычислять на любой итерации алгоритма при помощи двойственных цен, которые получаются во время выполнения стандартной версии алгоритма PHA. Приводятся результаты вычислительных экспериментов для стохастической задачи ввода узла в эксплуатацию и стохастической задачи оптимального расположения сервера, исследуется связь между ключевыми параметрами алгоритма PHA и качеством получаемых оценок снизу.

В [181] для задач стохастического смешанного целочисленного программирования авторы анализируют алгоритм двойственной декомпозиции Каро и Шульца с точки зрения возможностей параллельного решения. Предлагается формулировка задачи, которая позволяет устранить существенное препятствие для эффективного распараллеливания главной задачи, для чего используются решатели, основанные на методах внутренней точки, учитывающие структуру задачи. Результаты статьи демонстрируют высокий потенциал для ускорения с помощью распараллеливания и важность регуляризации (стабилизации) для задач двойственной оптимизации. Неравномерность распределения нагрузки остается единственным препятствием для оптимальной масштабируемости подзадач.

В статье [182] рассматривается класс двухэтапных задач стохастического целочисленного программирования с общими целочисленными переменными в обоих этапах и конечным числом реализаций неопределенных параметров. Основываясь на методе Бендерса, авторы предлагают алгоритм декомпозиции, который использует отсечения Гомори в обоих этапах. Отсечения Гомори для подзадач второго этапа параметризованы переменными из первого этапа, то есть они подходят для любых допустимых решений задач первого этапа. Кроме того, предлагается альтернативная реализация, которая использует метод Бендерса в методе ветвей и границ на первом этапе. Показана сходимость такого метода за конечное время. Также приводятся предварительные результаты

численных экспериментов с упрощенной реализацией данных алгоритмов, показывающие эффективность предложенных подходов.

В статье [183] изучается класс двухэтапных задач стохастической оптимизации с вероятностными ограничениями, где допустимое корректирующее решение второго этапа порождает дополнительную стоимость. Кроме того, авторы предлагают новую модель, где есть восстанавливающие решения, превращающие недопустимые решения в допустимые при помощи релаксации задачи второго этапа. Для решения задач этого класса разработаны алгоритмы декомпозиции со специальными отсечениями по оптимальности и допустимости. Вычислительные эксперименты для задачи планирования ресурсов с вероятностными ограничениями показывают, что предложенные алгоритмы предлагают высоко эффективные решения, по сравнению с подходом на основе смешанного целочисленного программирования и наивной декомпозиции.

Задачи распределения большой размерности обычно технически характеризуются как NP-трудные, что означает, что существующими методами невозможно найти оптимальное решение за полиномиальное время. Для решения задач смешанного целочисленного программирования большой размерности обычно рекомендуется использовать метод Бендерса. В работе [184] авторы используют метод декомпозиции Бендерса для решения задачи доставки бетона с заводов (Ready Mixed Concrete Dispatching Problem, RMCDP). Метод Бендерса включает в себя разделение исходной задачи доставки бетона на главную задачу (с оценкой снизу) и подзадачи (с оценками сверху). Главная задача использует целочисленные переменные, а подзадачи обычно являются задачами линейного программирования. Отсечения оптимальности Бендерса и отсечения допустимости Бендерса добавляются в главную задачу после каждой итерации решения подзадач. Предложенный метод проверен на примере реальной задачи, представлены результаты исследования.

В работе [185] описывается алгоритм кросс-декомпозиции, который сочетает метод Бендерса и сценарно–зависимую Лагранжеву декомпозицию для решения двухэтапной постановки стохастического программирования в задаче планирования инвестиций с полной компенсацией, где переменные первого этапа являются смешанными целочисленными, а переменные второго этапа — непрерывными. Алгоритм относится к новому классу кросс–декомпозиционных схем, полностью объединяет прямую и двойственную информацию через прямодвой-

ственные мульти–отсечения добавленные к алгоритму Бендерса и Лагранжевым координирующим задачам для каждого сценария. Потенциальные преимущества этой схемы кросс–декомпозиции показаны при помощи численных экспериментов для ряда экземпляров задачи оптимального местоположения в присутствии сбоев. В исходной постановке, где нижележащая релаксация линейного программирования является слабой, предложенный метод превосходит метод Бендерса с мульти–отсечениями. Если постановку задачи усилить дополнительными, более сильными ограничениями, эффективность обоих методов увеличивается, но метод кросс–декомпозиции однозначно остается лучшим для задач большой размерности.

Динамическое ценообразование уже стало типичной формой тарифа на электрическую энергию, где цена электрической энергии зависит в реальном времени от сложившихся отношений между спросом и предложением. Таким образом, для максимизации выгоды от динамического ценообразования, требуется оптимизация производственных процессов, учитывающая периоды низкой цены. В случае сетей водоснабжения, стоимость энергии расходуемой насосами вносит существенный вклад в конечные расходы, и важной задачей является оптимизация расписания работы насосных станций, чтобы согласовать расписание с изменяющейся ценой электроэнергии, при условии сохранения непрерывности подачи воды. В работе [186] рассматривается постановка задачи смешанного целочисленного нелинейного программирования для оптимизации расписания работы насосных станций. По причине нелинейности, большого типичного размера сетей водоснабжения и дискретности горизонта планирования проблема не может быть решена за разумное время при помощи стандартного программного обеспечения для задач оптимизации. В данной работе предлагается подход на основе лагранжевой декомпозиции, который существенно использует структуру задачи , что приводит к меньшему размеру подзадач, которые решаются независимо. Лагранжева декомпозиция сочетается с алгоритмом поиска, основанным на симуляции, с улучшенной дискриминирующей способностью, который способен находит допустимые решения высокого качества. Предложенный подход обнаруживает решения с гарантированными верхними и нижними оценками. Эти решения сравниваются с решениями задачи смешанного линейного целочисленного программирования, который использует кусочно-линейную аппроксимацию нелинейных ограничений и находит гло-

бальное оптимальное решение релаксированной задачи. Численные эксперименты для двух случаев реальных сетей водоснабжения показывают существенные снижение затрат при оптимизации расписания работы насосных станций.

2.2 Графовая интерпретация правил исключения

В теории локальных алгоритмов Журавлёва выделены две основные теоремы — теорема единственности и теорема мажорантности [187].

В теореме единственности утверждается, что результат вычисления основных предикатов локального алгоритма с монотонными функциями (которые не возрастают/ не убывают на заданном множестве) не зависит от порядка рассмотрения элементов множества. То есть, каком бы порядке мы не элиминировали переменные или подмножества переменных в задаче, оптимальное решение будет получено. В теореме мажорантности доказывается существование для всякого класса локально равных алгоритмов с одинаковой памятью наилучшего (мажорантного) локального алгоритма, т.е. алгоритма, который по заданной фиксированной системе окрестностей вычисляет заданные основные предикаты при фиксированных вспомогательных предикатах всегда, когда это делает любой другой алгоритм из рассматриваемого класса. То есть от того, в каком порядке мы элиминируем переменные или подмножества переменных зависит скорость нахождения оптимального решения. О.А. Щербина показал, что доказательство этой теоремы имеет неконструктивный характер [81], т.е. не позволяет осуществить прямое построение эффективного наилучшего алгоритма. В его диссертационной работе [81] сформулирован критерий существования оптимального порядка элиминации, который заключается в следующем.

Теорема 2.1. Для графа взаимосвязей существует оптимальный порядок элиминации тогда и только тогда, когда существует дерево декомпозиции, каждая вершина которого является кликой в соответствующем графе ограничений.

Порядок элиминации α — это выбор порядка решения подзадач. Теорема 2.1 определяет, в каком случае существует такой порядок решения подзадач, при котором работа локального элиминационного алгоритма будет оптимальной. Понятие дерева декомпозиции (ДД) введено Robertson и Seymour [188] для оценки подобия графа дереву. Выделение БД–структуры, которое рассматривалось в главе 1, является частным случаем древовидной декомпозиции [81]. Введём

необходимые понятия для дальнейшего изложения. Рассмотрим определение полного графа из [141].

Определение 2.3. Полным графом называется граф, каждая вершина которого связана с любой другой вершиной.

Введём понятие клики, определённое в [141].

Определение 2.4. Клика графа — это подмножество вершин X' графа G, в котором любые две вершины смежны, т.е. порожденный им подграф $G \mid X'$ является полным.

Далее рассмотрим определение симплициальной вершины из [141]

Определение 2.5. Симплициальной вершиной графа называется вершина, которая образует клику с соседними вершинами.

Наконец, исследуем понятие дерева клик, введённое в [81].

Определение 2.6. Деревом клик графа G будем называть дерево, для которого вершинами $k \in K$ являются максимальные клики графа G, такие что для каждой пары k', k'' максимальных клик в G клика $k' \cap k''$ содержится в каждой максимальной клике из (однозначно определенного) k'–k"–пути в B.

Сформулируем альтернативное определение для дерева клик.

Определение 2.7. Деревом клик графа G называется пара $(\{\mathbf{K_i}|i \in I\})$, $B = (I, F))$, где $\{\mathbf{K_i}|i \in I\}$ — максимальные клики графа G и B — дерево с множеством вершин i и множеством ребер $F \subseteq I \times I$ такими, что:

1) $\bigcup_{i \in I} \mathbf{K_i} = K$, где K — множество максимальных клик графа G;

2) для всех ребер $(v, w) \in E$ существует $i \in I$ такое, что $v \in \mathbf{K_i}$, и $w \in \mathbf{K_i}$;

3) для всех пар K', K'' максимальных клик в G клика $K' \cap K''$ содержится в каждой максимальной клике из (однозначно определенного) K'–K"–пути в B.

Если построить дерево клик для графа ограничений некоторой задачи ДО, то каждая вершина этого дерева будет кликой для соответствующего графа

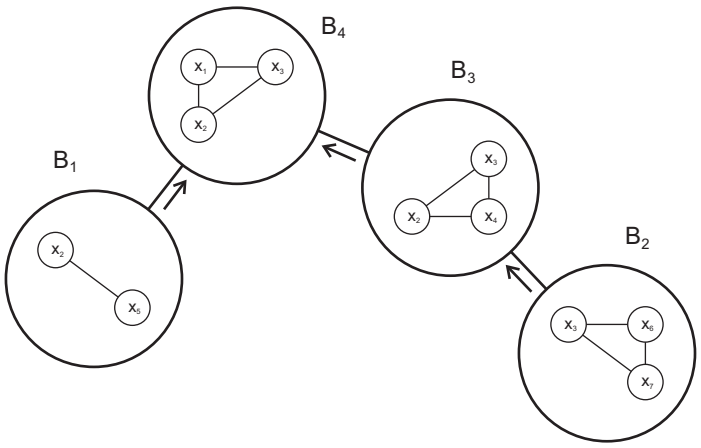


Рисунок 2.2: Дерево клик для задачи ДО из примера (2.8)

ограничений. Согласно теореме 2.1 необходимо, чтобы дерево клик также являлось и деревом декомпозиции задачи ДО.

Древовидная декомпозиция — это метод сжатия подмножеств переменных в супер–переменные таким образом, что полученная супер–задача ДО имеет ациклическую форму в представлении в виде двойственного графа. Полученный граф имеет структуру дерева и называется деревом декомпозиции. Перейдём к определению дерева декомпозиции из [81].

Определение 2.8. Деревом декомпозиции (ДД) для заданного графа $G(X, E)$ называется пара $(\{\mathbf{X_i}|i \in I\}), T = (I, F))$, где $\{\mathbf{X_i}|i \in I\}$ — семейство подмножеств $v \in X$ и T — дерево с множеством вершин i и множеством ребер $F \subseteq I \times I$ такими, что:

1) $\bigcup_{i \in I} \mathbf{X_i} = X$;

2) для всех ребер $(v, w) \in E$ существует $i \in I$ такое, что $v \in \mathbf{X_i}$, и $w \in \mathbf{X_i}$;

3) для всех $i, j, k \in I$ таких, что j лежит на пути T из i в k, справедливо включение $\mathbf{X_i} \cap \mathbf{X_k} \subseteq \mathbf{X_j}$.

Далее рассмотрим свойства ДД [81]:

— Каждая переменная из первоначальной задачи появляется по меньшей мере в одной из подзадач.

— Если две переменные первоначальной задачи связаны ограничением, то должны появиться вместе (наряду с этим ограничением) по меньшей мере в одной из подзадач.

— Если какая–то переменная появляется в двух подзадачах в дереве, то должна появляться в каждой подзадаче вдоль пути, соединяющего эти подзадачи.

Построим ДД для задачи ДО из примера (2.8). Согласно определению ДД, $\{\mathbf{X_i}|i \in I\}$ — семейство подмножеств $v = \{\mathbf{x}_1,, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$ (см. рис. 2.3б), а T — дерево с множеством вершин $i = \{1; 2; 3; 4\}$ и множеством ребер $F = \{(1, 2); (2, 4); (3, 4)\}$. При этом выполняются следующие условия:

1) $\mathbf{x}_1 \cup \mathbf{x}_2 \cup \mathbf{x}_3 \cup \mathbf{x}_4 = X$,

2) для всех ребер $E = \{(x_1, x_2); (x_1, x_3); (x_2, x_3); (x_2, x_4); (x_2, x_5); (x_3, x_4); (x_3, x_6); (x$ существует $i \in I$ такое, что $v \in \mathbf{X}_i$, и $w \in \mathbf{X}_i$

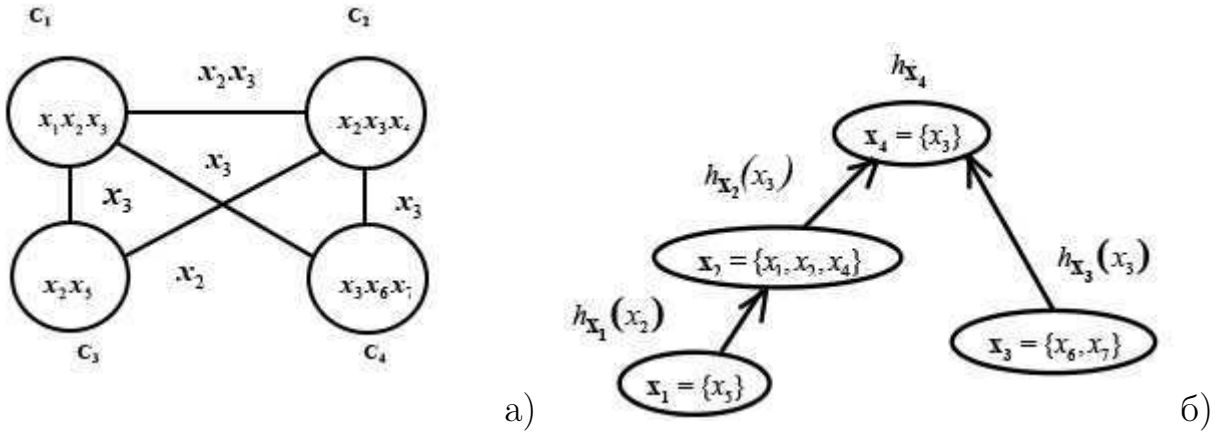
3) для $i = 1$,$j = 2$,а $k = 4$ таких, что $j = 2$ лежит на пути T из $i = 1$ в $k = 4$, справедливо включение $\mathbf{x}_1 \cap \mathbf{x}_4 = \{x_2\} \subseteq \mathbf{x}_2 = \{x_1, x_2, x_4\}$; в данном дереве три вершины встречаются только для $\mathbf{x}_1$, $\mathbf{x}_2$ и $\mathbf{x}_4$.

Полученная ДД для задачи ДО из примера (2.8) соответствует поиску симплициальных вершин и соответствующих максимальных клик, то есть дереву клик (рис. 2.2). Далее установим, каким образом могут быть связаны дерево декомпозиции и дерево клик, чтобы понять, в каких условиях выполняются условия теоремы 2.1. Дерево клик является частным случаем дерева декомпозиции. Таким образом, для того, чтобы выполнялось условия теоремы 2.1, необходимо, чтобы каждая вершина дерева декомпозиции была кликой. Таким образом, ДД можно выделить с помощью алгоритма поиска максимальных клик в графе. Для этого существует целый класс ПО, в частности, с открытым кодом. Например, NetworkX даёт приближённое решение этой задачи (процедура max_clique), в maxClique заложены точные алгоритмы, а с помощью OpenOpt можно получить точные и приближённые решения, при этом можно управлять процессом: есть возможность указать рёбра, которые должны быть включены. В данной работе алгоритм поиска максимальных клик MCS [189] используется для построения порядка элиминации. Но сама задача о клике является NP–трудной задачей. Рассмотрим теорему о взаимосвязи стягивающего дерева и дерева клик из [142].

Теорема 2.2. Дерево клик — максимальное стягивающее дерево для двойственного графа.

Таким образом, дерево клик также возможно построить с помощью стягивающего дерева для двойственного графа задачи ДО. Стягивающее дерево для данного неориентированного графа — суграф (т.е. часть графа, имеющая то же множество вершин, что и сам граф [140]) в виде дерева.

На рисунке 2.3 для задачи ДО из примера (2.8) продемонстрирован двойственный граф и соответствующее ему максимальное стягивающее дерево. Можно видеть, что для того, чтобы получить максимальное стягивающее дерево, из двойственного графа пришлось удалить ребра из C_1 в C_3 и из C_2 в C_4. Максимальным стягивающим деревом для двойственного графа является



а) б)

Рисунок 2.3: Иллюстрация теоремы о взаимосвязи стягивающего дерева и дерева клик для задачи ДО из примера (2.8): а) двойственный граф, б) максимальное стягивающее дерево

элиминационное дерево [81], рассматриваемое далее.

Термин «элиминационная игра» впервые введён Партером [118] в 1961 году как интерпретация метода исключения Гаусса на графах. Рассмотрим определение элиминационного графа из [81]

Определение 2.9. Граф, полученный из графа взаимосвязей переменных с помощью удаления некоторой вершины x_k и всех ребер, исходящих из нее, а затем соединения ребрами всех ранее не соседних вершин в окрестности x_k, называется x_k — *элиминационным графом* G^k. Описанная операция называется *элиминацией вершины* x_k. Последовательность всех элиминированных вершин называется *порядок элиминации* α.

Замечание. Под понятием «элиминационная игра» будем иметь ввиду процесс последовательной элиминации вершин $x_1, \ldots, x_n$, которая порождает последовательность графов $G = G^0, G^1, G^2, \ldots, G^n$.

Порядок элиминации для конкретной задачи ДО неоднозначен. Для задачи ДО из примера (2.8) предлагается следующий порядок элиминации: $\alpha = \{x_1, x_5, \{x_2, x_4\}, \{x_6, x_7\}, x_3\}$. Для задачи ДО из примера (2.8) «элиминационная игра» может выглядеть следующим образом (рис. 2.4). Первым в данной последовательности будет граф ограничений задачи ДО $G^0 = G$. Смотрим на порядок элиминации $\alpha = \{x_1, x_5, \{x_2, x_4\}, \{x_6, x_7\}, x_3\}$ и видим, что первой вершиной по порядку является вершина x_1. Исключим x_1 из графа G^0 и получим граф G^1. Снова смотрим на порядок элиминации и исключаем x_5. Получим следующий в последовательности граф — G^2. Далее согласно порядку элиминации исключим блок переменных $\{x_2, x_4\}$ и получим граф G^3. Аналогичным образом исключим блок $\{x_6, x_7\}$ и получим последний граф последовательности G^5, состоящий из одной вершины x_3. На заключительном шаге исключим оставшуюся переменную x_3. «Элиминационная игра» закончена. Обозначим процеду-

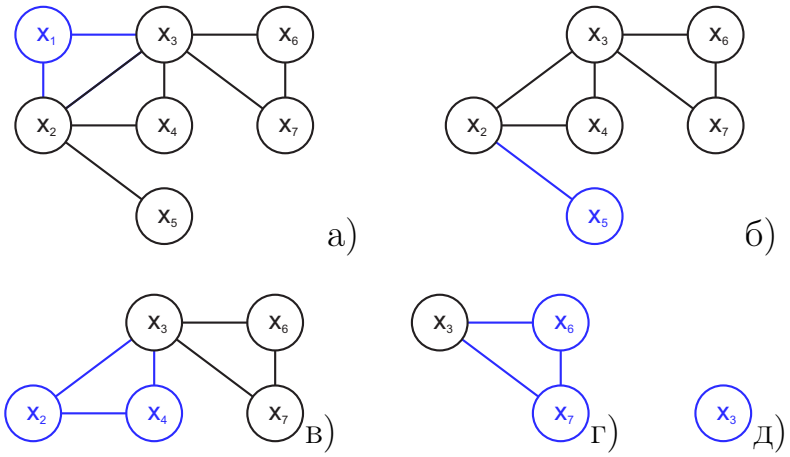


Рисунок 2.4: «Элиминационная игра» для задачи ДО из примера (2.8): а)G^0; б)G^1; в)G^2; г)G^3; д)G^4

ру элиминации $G' = elim\{x_i\}$. Это означает, что из графа G' удалена вершина x_i, а все элементы её окрестности, которые не были соединены ребром теперь являются соседями. Для того, чтобы последовательно элиминировать все вершины графа $G(X, E)$ воспользуемся **алгоритмом элиминационной игры**. Введём понятие монотонной окрестности для контекста «элиминационной игры» [81]. Для данного порядка элиминации α вершин графа G в виде $x_1, \ldots, x_n$ через $\overline{\alpha}_i$ обозначим множество вершин с индексами из α, большими $i-1$: $\overline{\alpha}_i = \{x_i, x_{i+1}, \ldots, x_n\}$.

Алгоритм 3 Алгоритм элиминационной игры [81]

Шаг 1. Выбираем первую по порядку элиминации вершину.

Шаг 2. Добавляем, если нужно, необходимые ребра так, чтобы данная вершина и все её соседние вершины образовывали клику.

Шаг 3. Удаляем вершину из измененного графа.

Шаг 4. Продолжаем, пока не пройдём все вершины.

Шаг 5. Добавляем в исходном графе все добавленные на каждом шаге ребра.

Определение 2.10. Монотонной окрестностью вершины x_i будем называть множество вершин, соседних с x_i, с индексами, большими, чем i (согласно порядку α): $\overline{Nb}^{\alpha}_{G}(x_i) = \{x_j \in Nb_G(x_i) | j > i\} = Nb_G \cap \overline{\alpha}$.

Введём определение пополненного графа из [81]

Определение 2.11. Пополненным графом G^{+}_{α} называется граф, который получается из графа ограничений в результате работы алгоритма элиминационной игры 3.

Замечание. Отметим, что G^{+}_{α} может быть построен следующим образом. Пусть α' будет порядком элиминации G', полученным из α при удалении v_1. Возьмём граф G', полученный при элиминации v_1. Построим рекурсивно $G'^{+}_{\alpha'}$, добавляя в него на каждом шаге рекурсии i вершину v_i и ее инцидентные рёбра.

Рассмотрим определение элиминационного дерево, введённое в [81].

Определение 2.12. Элиминационным деревом (ЭД) графа $G(X, E)$ для упорядочения вершин α называется ориентированное дерево $\vec{T}_{\alpha}$, имеющее то же множество вершин X, что и исходный граф G, а множество ребер ЭД определяется с помощью отношения «предок — потомок» следующим образом: предком вершины x является первая (согласно упорядочению α) вершина из монотонной окрестности $\overline{Nb}^{\alpha}_{G^{+}_{\alpha}}(x)$ вершины x в пополненном графе G^{+}_{α}.

Введём понятие обобщённого ЭД.

Определение 2.13. Обобщённое дерево элиминаций (ОЭД) — такое ЭД, каждой вершиной которого является супер–вершина исходного графа.

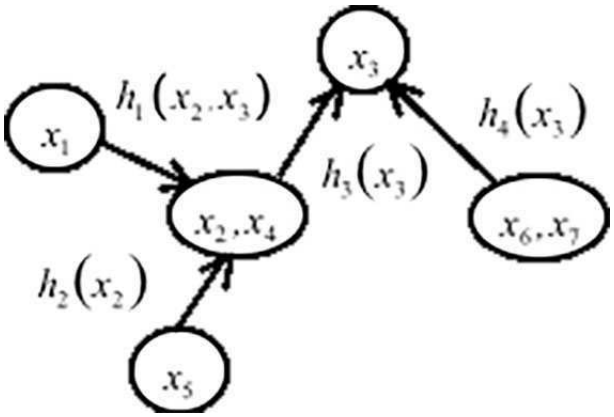


Рисунок 2.5: Элиминационное дерево, описывающее задачу ДО из примера (2.8)

Замечание. Очевидно, что обобщённое ЭД можно получить путём объединения поддеревьев данного ЭД.

С учётом алгоритма элиминационной игры сформулируем алгоритм для выделения обобщённого ЭД. Поясним алгоритм 4. Пока в графе остаётся хотя

Алгоритм 4 Алгоритм выделения обобщённого ЭД

УТВЕРЖДЕНИЕ $G'(X', E') := G(X, E)$, $F := \emptyset$; ПРОЦЕДУРА $ED(G(X, E))$:

Шаг 1. ЕСЛИ $G'(X', E') = \emptyset$ ТО КОНЕЦ ПРОЦЕДУРЫ

Шаг 2. ЕСЛИ $Nb(x_{i_1}) \cap \ldots \cap Nb(x_{i_k}) = \emptyset$ ТО $G'(X', E') = elim(\{x_{i_1}, \ldots, x_{i_k}\})$

Шаг 3. ЕСЛИ $x_i \in Nb(x_j)$ ТО $F = F \cup (x_i, x_j)$, где x_j — супер–вершина, элиминированная на предыдущей итерации шаге, а x_i — супер–вершина, элиминированная на текущей итерации.

Шаг 4. $ED(G'(X', E'))$

бы одна супер–вершина, процедура будет работать. На каждой итерации будут элиминированы все супер–вершины, окрестности которых не пересекаются. Рёбрами данного дерева будут все рёбра, связывающие элиминированные супер–вершины на текущей и предыдущей итерации процедуры если на предыдущей итерации они были соседями в графе G'. Полученное дерево является ОЭД по определению. Перейдём к теореме о соответствии дерева декомпозиции и дерева элиминации [81].

Теорема 2.3. Обобщённое элиминационное дерево задачи ДО соответствует дереву её декомпозиции.

Сформулируем и докажем альтернативную теорему для более точного понятия взаимосвязи для данных структур.

Теорема 2.4. Пусть $G = (X, E)$ — заданный граф, α — порядок элиминации элементов графа G, G^+_α — пополненный граф относительно G. Для данных G и порядка элиминации α алгоритм 4 строит древовидную декомпозицию.

Доказательство. Пусть для определённости $X = \{x_1, \ldots, x_n\}$, и для всех $x_i \in X$ $\alpha(x_i) = i$. Рассмотрим алгоритм 4. Для графа, состоящего из одной вершины решение тривиально. В противном случае, докажем каждый пункт из определения дерева декомпозиции.

1) Поскольку все вершины дерева элиминации — это вершины, полученные при элиминации графа $G(X, E)$, их объединение совпадает с множеством вершин исходного графа, то есть $\bigcup_{i \in I} \mathbf{X_i} = X$;

2) Каждое из рёбер исходного графа $(v, w) \in E$ соединяет вершину v исходного графа и переменную из её окрестности $\{w\} \in Nb(v)$. Поскольку ребро в дереве элиминации на каждой итерации алгоритма 4 соединяет каждую элиминируемую вершину v и её окрестность $Nb(v)$, то для всех ребер $(v, w) \in E$ существует $i \in I$ такое, что $v \in \mathbf{X_i}$, и $w \in \mathbf{X_i}$, где $\mathbf{X_i} = Nb(v) \cup \{v\}$;

3) Пусть среди множества итераций процедуры в алгоритма 4 существуют итерации с номерами i, j, k, которые совершаются алгоритмом в заданной последовательности. Тогда ребро в дереве элиминации соединяет каждую элиминируемую вершину v_i и $\{v_j\} \in Nb(v_i)$ на j–той итерации алгоритма, а затем каждую элиминируемую вершину v_j и $\{v_k\} \in Nb(v_j)$ на k–той итерации. Предположим, что утверждение $\mathbf{X_i} \cap \mathbf{X_k} \subseteq \mathbf{X_j}$, где $\mathbf{X_i} = \{v_{i-1}\} \in Nb(v_i)$, $\mathbf{X_j} = \{v_{j-1}\} \in Nb(v_j)$, $\mathbf{X_k} = \{v_{k-1}\} \in Nb(v_k)$, не справедливо. Значит имеет место $\mathbf{X_i} \cap \mathbf{X_k} \subsetneq \mathbf{X_j}$ и в пересечении $\mathbf{X_i} \cap \mathbf{X_k}$ существует некоторый элемент $x_s \notin \mathbf{X_j}$. Значит $x_s \in \mathbf{X_i}$ и $x_s \in \mathbf{X_k}$. Поскольку элемент x_s находится в $\mathbf{X_k}$, значит он не был элиминирован на более ранних итерациях, в том числе и на итерации $\mathbf{X_j}$. Значит $x_s \in \mathbf{X_j}$ и мы пришли к противоречию. Таким образом для всех $i, j, k \in I$ таких, что j лежит на пути T из i в k, справедливо включение $\mathbf{X_i} \cap \mathbf{X_k} \subseteq \mathbf{X_j}$, где $\mathbf{X_i} = \{v_{i-1}\} \in Nb(v_i)$.

Теорема доказана.

На основе теоремы 2.4 получены окончательные результаты в работе О.А. Щербины [142], которые позволяют установить взаимосвязь между ДД задачи ДО и ОЭД, которое является бесконтурным орграфом вычислительной процедуры локального элиминационного алгоритма (ЛЭА) (см. [81]). Особенностью ЛЭА является использование локальной информации об окрестностях так называемых связывающих переменных, то есть переменных, принадлежащих одновременно нескольким окрестностям. Поэтому ЛЭА позволяет получать глобальное решение задачи с помощью локальных вычислений, которые обычно являются решениями соответствующих подзадач. ЛЭА в общем виде представляется как последовательное исключение переменных с сохранением информации о них. Подробнее ЛЭА рассматривается в следующей главе.

Блоки являются частным случаем разбиений, они не пересекаются и образуют разбиение графа. Если задача ДО разбита на блоки, соответствующие подмножествам переменных (называемых супер–переменными), полученная блочная структура может описываться с помощью структурного конденсированного графа, супер–вершины которого соответствуют подмножествам переменных (или блокам) исходного графа, а супер–ребра соответствуют соседним блокам. Использование метода сжатия подмножеств переменных в супер–переменные позволяет получить конденсированные или супер–задачи ДО, имеющие более простую структуру, которые могут быть решены более эффективно, например, древовидную. Таким образом, схема блочной элиминации является элиминационной процедурой, в которой вершины каждого блока элиминируются одновременно, кластером [190].

Из теоремы 2.4 следует, что для определения ДД достаточно найти ОЭД, соответствующее графу ограничений задачи ДО. Таким образом, методы, используемые для нахождения порядка элиминации помогают разбить исходную задачу на подзадачи, где каждая подзадача соответствует элиминируемой переменной или группе переменных. Далее рассмотрим эвристики для определения порядка элиминации в задачах ДО, а также сравним их между собой.

2.3 Численные тесты выбора порядка исключения переменных

Задача поиска оптимального упорядочения является NP–полной [191], поэтому на практике для нахождения элиминационной последовательности переменных используются всевозможные эвристики. Для задач ДО со специальной структурой использование эвристик как алгоритмов упорядочивания супер–переменных для последующей элиминации представляет существенный интерес. Рассмотрим подробнее наиболее известные эвристики.

Алгоритм упорядочивания минимальной степени MD (Minimum Degree) является одной из наиболее широко используемых эвристик, т.к. он дает хорошие результаты с относительно небольшим пополнением для разреженных графов.

В алгоритме MD выбирается вершина v графа G с минимальной степенью. Далее строится граф G', получаемый путем создания клики из вершины v и ее соседей с последующим удалением v и инцидентных ей ребер. Рекурсивно из G' с помощью эвристики создается *хордальный* суперграф H'. И, наконец, строится связный суперграф H из G, путем добавления v и инцидентных ей ребер из G к H'. Будучи алгоритмом локальной минимизации, алгоритм MD не всегда дает упорядочение с минимальным пополнением для графа в целом. Существуют модификации алгоритма минимальной степени – алгоритмы MMD (Multiple Minimum Degree [192]) и AMD (Approximate Minimum Degree) [193].

Алгоритм рекурсивного разбиения ND (Nested Dissection) [194] — глобальный эвристический рекуррентный алгоритм, находящий сепаратор, т.е. множество вершин S, разделяющее граф на две части A и B, помещая его в упорядочении последним. Алгоритм применяется рекуррентно к частям графа A и B, пока их размеры не станут меньше, чем некоторое пороговое значение.

Существуют также гибридные схемы методов рекурсивного разбиения ND и минимальной степени MD [148], [195], [149].

В [189] был предложен алгоритм MCS (Maximum Cardinality Search — поиск по максимальной степени). Алгоритм MCS на некотором графе G производит за время $O(n+m)$ полное упорядочение множества вершин следующим образом. Из произвольной вершины выбирается любая, еще не пронумерованная вершина, смежная максимальному числу уже пронумерованных вершин.

Эвристика минимального пополнения MIN-FILL (Minimum Fill-in) [196] работает практически так же, как и описанная выше эвристика минимальной степени MD, с той лишь разницей, что здесь на каждом шаге выбирается такая вершина, чтобы число добавляемых к ней ребер, необходимых для получения клики, было минимальным.

В работе [197] предложен алгоритм, вычисляющий хорошую элиминационную последовательность за линейное время, который был назван лексикографический поиск в ширину LEX–BFS (Lexicographic Breadth–First Search). Суть данного метода заключается в следующем. Вершины нумеруются от n до 1 (нумерация фиксирует позиции переменных) в упорядочении. Далее, для каждой вершины создается метка, содержащая множество чисел, записанных по убыванию. Таким образом вершины могут быть лексикографически упорядочены согласно их меткам.

Эффективность вышеназванных эвристик определим с помощью вычислительного эксперимента для исследования влияния эвристик на время решения задач ДО. Целью данного эксперимента является исследование влияния каждого из пяти алгоритмов упорядочивания переменных на работу локального элиминационного алгоритма. Для данного эксперимента использовались следующие эвристики, описанные выше:

— Алгоритм минимальной степени MD

— Алгоритм рекурсивного разбиения ND

— Алгоритм поиска по максимальной степени MCS

— Алгоритм минимального пополнения MIN–FILL

— Алгоритм лексикографического поиска в ширину LEX–BFS

Тестовые задачи ДО генерировались на основе уже существующих матриц из библиотеки CSP [1]. Указанная библиотека содержит различные классы матриц для задач удовлетворения ограничений, среди которых были реальные задачи для приложений в промышленности (DaimlerChrysler, NASA, ISCAS) и искусственно созданные примеры задач.

[1] http://faculty.cse.tamu.edu/davis/welcome.html

Тестовые задачи ДО строились следующим образом. Структура ограничений линейной задачи ДО с бинарными переменными задавались с помощью гиперграфов из библиотеки CSP. Для построения ограничения i бралось очередное гиперребро гиперграфа, содержащее множество переменных X_{S_i}, входящих в строящееся ограничение. Далее с помощью процедуры, использующей датчик случайных чисел, строились коэффициенты A_{S_i} при соответствующих переменных, тогда левая часть i-го ограничения имела вид $A_{S_i}X_{S_i}$. Правая часть i-го ограничения имеет вид $\sigma \sum A_{S_i}$, где σ – случайное число из интервала (0, 1). Целевая функция линейна, содержит все переменные – вершины гиперграфа, причем коэффициенты c_j целевой функции $\sum_{j=1}^{n} c_j x_j \to \max$ строились с помощью процедуры, использующей датчик случайных чисел.

Далее для каждого графа взаимосвязей применялись алгоритмы упорядочивания вершин в графе взаимосвязей, а именно MD, ND, MCS, MIN-FILL и LEX-BFS. В данном эксперименте был выбран пакет MATLAB [2], так как версии вышеназванных алгоритмов реализованы в этой среде в виде стандартных функций. Затем пример задачи и полученный порядок элиминации подгружался в AMPL, где итоговая задача решалась с помощью ЛЭА.

В данном эксперименте всего было взято 150 примеров случайно выбранных задач. Время решения одних и тех же задач с разными эвристиками сравнивалось и выбиралась лучшая из них. Таким образом, для алгоритма ND минимальное время работы алгоритма не было достигнуто, для MD – 7 раз, LEX-BFS – 11 раз, MCS – 39 раз и MIN-FILL – 93 раза (рис. 2.6).

Основным результатом данного эксперимента является исследование влияния пяти алгоритмов упорядочивания вершин на время решения разреженных задач ДО с помощью локального элиминационного алгоритма. В результате проведенного вычислительного эксперимента было выявлено, что порядок переменных оказывает значительное влияние на время решения задачи. Кроме того, полученный порядок элиминации используется в декомпозиции для построения БД–структуры задачи ДО. Локальный элиминационный алгоритм для одних БД–структур получает оптимальное решение быстрее, а для других — медленнее. Скорость работы локального элиминационного алгоритма связана с тем, каким образом получена из графа взаимосвязей та или иная структура задачи ДО. При этом полученные БД–структуры различались порядком элиминации

[2] http://matlab.ru

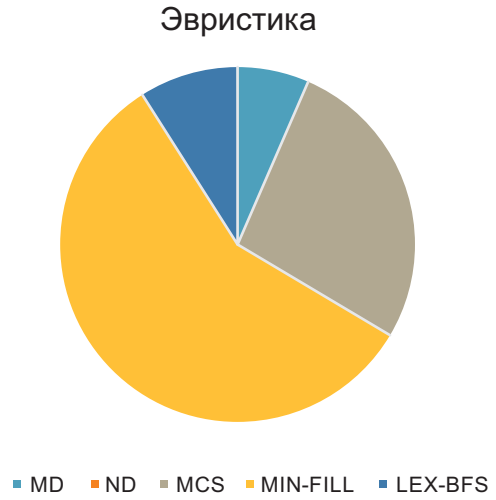


Рисунок 2.6: Сравнение эвристик по минимальному времени работы локального элиминационного алгоритма с полученным порядком элиминации

переменных в графе: лучшие результаты получились с эвристикой MIN–FILL, а худшие — с эвристикой ND.

Глава 3

Тестирование и распараллеливание задач квазиблочной структуры

В данной главе рассматриваются локальные элиминационные алгоритмы О.А. Щербины [81] в применении к задачам с квазиблочной структурой — локальные блочно–элиминационные алгоритмы (ЛБЭА). Рассматриваются модификации ЛБЭА, которые позволяют существенно его ускорить. Это эвристический алгоритм (ЭЛБЭА), а также ЛБЭА, использующие предобработку, параметрическую оптимизацию и релаксации. Также осуществляется распараллеливание задач с квазиблочной структурой. Для этого используется независимое решение промежуточных блочных задач на отдельных процессорах. Приводится обзор распараллеливания задач ЦЛП.

3.1 Локальный блочно–элиминационный алгоритм

Принцип работы локальных алгоритмов заключается в следующем. Вначале вычисляется информация о локальных элементах структуры задачи, которая записывается в виде новых зависимостей, добавляемых к задаче. Затем просмотренные элементы и использованные зависимости исключаются из процесса вычисления [74, 142].Когда все возможные элементы исключены, восстанавливается глобальное решение исходя из зафиксированной ранее локальной информации на каждом шаге алгоритма. Рассмотрим подробнее эти процессы.

Основная идея ЛЭА состоит в последовательном исключении переменных с сохранением информации об этих переменных. Процедура ЛЭА разбивается на две части:

— элиминация элементов, вычисление и запоминание информации в виде локальных решений и получение в конце значения критерия;

— нахождение глобального решения всей задачи по найденным в первой части таблицам с локальными решениями, обеспечивающего в первой части достижение критерия.

Первая часть ЛЭА состоит из нескольких этапов:

Шаг 1. Определение порядка исключения переменных согласно некоторой эвристике.

Шаг 2. Запись локальной информации об этих переменных в виде новых зависимостей, добавляемых к задаче.

Шаг 3. Исключение просмотренных элементов и использованных зависимостей согласно порядку исключения переменных.

Алгоритмическая схема ЛЭА представляет собой бесконтурный орграф, вершины которого соответствуют локальным подзадачам, а ребра — выражают информационную (или концептуальную) зависимость подзадач друг от друга. Проиллюстрируем работу частного случая ЛЭА, а именно его использование для задач с БД–структурой — ЛБЭА.

Изложим ЛБЭА для решения задач с БД–структурой (2.5)–(2.7). Введём $S_{rr'}$ — множество индексов переменных, принадлежащих одновременно блокам B_r и $B_{r'}$. Если множество индексов переменных $S = \{j_1, \ldots, j_q\}$, то вектор переменных будет таким: $X_S = (x_{j_1}, \ldots, x_{j_q})$. X_{S_r} — вектор переменных, принадлежащих блоку B_r, $X_{S_{rr'}}$ — вектор переменных, общих для блоков B_r и $B_{r'}$. Обозначим через z_{D_r} следующую задачу: для каждого вектора $X_{S_{prr}}$ найти X_{S_r} и $X_{S_{rr'}}$, такие, чтобы

$$f_{D_r}(X_{S_{prr}}) = \max\{C_{S_r}X_{S_r} + \sum_{r' \in J_r}[f_{D_{r'}}(X_{S_{rr'}}) + C_{S_{rr'}}X_{S_{rr'}}]\}$$

при ограничениях

$$A_{S_r}X_{S_r} \le b_r - \sum_{r' \in J_r} A_{S_{rr'}}X_{S_{rr'}} - A_{S_{prr}}X_{S_{prr}}.$$

Здесь $f_{D_{r'}}(X_{S_{rr'}})$ — значение целевой функции задачи z_{D_r}, соответствующей дереву D_r. Решение задачи z_{D_r} для вершины r дерева D при фиксированном

векторе $X_{S_{prr}}$ обозначим таким образом:

$$X_{D_r}(X_{S_{prr}}) = \bigcup_{r' \in J_r} [X_{D_{r'}}(X_{S_{rr'}}) \bigcup X_{S_{rr'}}] \bigcup X_{S_r}.$$

Понятно, что $f_{D_{r'}}(X_{S_{rr'}}) = C_{D_{r'}} X_{D_{r'}}$. Нетрудно заметить, что если зафиксировать вектор $X_{S_{prr}}$, то задача (2.5)–(2.7) распадается на две задачи: первая соответствует дереву D_r; а вторая — $D \setminus D_r$. На этом свойстве и основано применение ЛЭА для решения задач ДО с БД–структурой. Алгоритм вычисляет информацию, поднимаясь от «листьев дерева к корневой вершине». Применение ЛБЭА к задаче ДО, характеризующейся деревом D инцидентности блоков с k вершинами, состоящим из L слоев, выглядит так. Пусть в ν–м слое имеется l_ν вершин $R_\nu = \{r_1^{(\nu)}, \ldots, r_{l_\nu}^{(\nu)}\}$. J_r — множества индексов переменных, которые являются общими для данного блока $r = r_{l_\nu}^{(\nu)}$ и блока $r' = r_{l_{\nu+1}}^{(\nu+1)}$, если множество $S_{rr'}$ не пусто. Другими словами $J_{p_r} = S_{p_r r}$. Рассмотрим шаги ЛБЭА.

Алгоритм 5 ЛБЭА для решения задач с БД–структурой

Шаг 1. Положить $\nu = L$, $J_r = \emptyset$ для всех $r \in R_L$.

Шаг 2. Для каждой вершины $r = r_l^{(\nu)}$, $l = 1, \ldots, l_\nu$ слоя ν дерева D решить задачу z_{D_r}. Если эта задача не имеет решения ни для одной вершины данного слоя — перейти к шагу 5, в противном случае — к шагу 3.

Шаг 3. Если $\nu \geq 2$, то перейти на слой выше, т.е. положить $\nu := \nu - 1$ и перейти к шагу 2, иначе — к шагу 4.

Шаг 4. Конец вычислений. Решение задачи z_{D_r} на уровне $\nu = 1$ является решением исходной задачи: $z_{max} = f_{D_1}$.

Шаг 5. Конец вычислений. Задача не имеет допустимых решений.

Поясним более подробно работу ЛБЭА. Пусть первоначально исходная задача является блочно–лестничной и содержит R блоков. Порядок исключения блоков — $\{r_1, r_2, \ldots, r_R\}$. Все блоки, кроме первого и последнего, будут содержать три вида переменных: блочная, сепаратор и наследственная. Наследственные переменные — это переменные, принадлежащие одновременно текущему блоку и предыдущему. Сепараторы для данной структуры задачи — это пере-

менные, принадлежащие одновременно текущему блоку и следующему. Рассмотрим переменные, соответствующие первому исключаемому блоку r_1: он содержит t_1 свободных переменных и s_1 сепараторов — переменных: которые принадлежат текущему блоку и следующему. Создадим подзадачу $P_1(X_1^s)$ и зафиксируем сепараторы первого блока $X_1^s = \{0, 0, \ldots, 0, 0\}$. Решим полученную подзадачу и занесём решение (X_1^s, X_1^t, Z_1) в таблицу ЛБЭА, где X_1^s — зафиксированное значение сепаратора, X_1^t — значение вектора свободных переменных и Z_1^s — экстремальное значение целевой функции. По аналогии решим остальные подзадачи для каждого значения сепаратора от $X_1^s = \{0, 0, \ldots, 0, 1\}$ до $X_1^s = \{1, 1, \ldots, 1, 1\}$, всего получится 2^{s_1} подзадач для данного блока. Отметим, что если для блока решена подзадача, и он связан со следующим блоком, то он будет являться потомком этого блока. В частности, первый блок будет являться потомком по отношению ко второму блоку.

Рассмотрим второй блок: он содержит s_2 сепараторов, t_2 свободных переменных и u_2 наследственных. Так как у первого и второго блока есть общие переменные, то $u_2 = s_1$. Создадим подзадачу $P_2(X_2^s)$ и зафиксируем сепараторы второго блока $X_2^s = \{0, 0, \ldots, 0, 0\}$. Наследственные переменные для задачи $P_2(X_2^s)$ представлены в виде табличной функции, соответствующей подзадаче $P_1(X_1^s)$: согласно значениям этих переменных X_1^s из таблицы выбирается соответствующее значение целевой функции Z_1 и прибавляется к значению целевой функции $Z_2 + Z_1$. Таким образом задача $P_2(X_2^s)$ решается относительно переменных t_2 и u_2 с функционалом $Z_2 + Z_1$. Решения задачи для всех значений сепаратора заносятся в соответствующую таблицу. Всего таких подзадач для первых двух блоков $2^{s_1} \cdot 2^{s_2}$.

По аналогии решим задачи, соответствующие всем остальным блокам кроме последнего. У последнего блока будет единственное решение $(None, X_R, Z_R)$, так как у него нет сепараторов. Переходим к восстановлению глобального решения задачи. Ищем значение вектора решений X_R в таблице блока $R-1$ среди значений сепараторов. Соответствующие строки будут содержать решения X_{R-1}^t. Если в одном блоке имеется несколько решений, выбираем оптимальное и расширяем вектор X_R за счёт полученных значений. Полученные вектора решений ищем в следующей таблице и так далее, пока блоки не закончатся. Вектор, достроенный в последнем блоке будет глобальным вектором решения исходной задачи.

Рассмотрим особенности БД–структуры. Пусть для задачи с БД–структурой определён порядок решения подзадач, соответствующих блокам. Тогда если у блоков нет потомков, подзадачи решаются аналогично подзадачам для БЛ–структуры. Если потомки есть, то его подзадача строится следующим образом. Рассмотрим блок r, он содержит s_r сепараторов, t_r свободных переменных и u_r наследственных. Пусть у него k потомков, тогда множество наследственных переменных разбивается на k подмножеств: $u_r = p_1 + p_2 + \ldots + p_k$. Значит создадим подзадачу $P_r(X_r^s)$ и зафиксируем сепараторы r–того блока $X_r^s = \{0, 0, \ldots, 0, 0\}$. Наследственные переменные для задачи $P_r(X_r^s)$ представлены в виде k табличных функций: согласно значениям этих переменных X_{r-1}^s из каждой i–той таблицы выбирается соответствующее значение целевой функции Z_{r-1}^i и прибавляется к значению целевой функции $Z_r = Z_r + \sum Z_i$, где $i = 1, \ldots, k$. Решения задачи для всех значений сепаратора заносятся в соответствующую таблицу. Всего таких подзадач для первых двух блоков $2^{s_1} \cdot 2^{s_{r-1}}$.

Подзадача, соответствующая последнему блоку R решается аналогично за исключением того, что имеет единственное решение ($None$, X_R, Z_R). Для восстановления глобального решения на каждом шаге необходимо искать значение вектора решений X_R в таблицах блоков–потомков блока R среди значений сепараторов. Соответствующие строки таблиц будут содержать решения подзадач, причём если в одном блоке имеется несколько решений, выбираем оптимальное из них и расширяем вектор X_R за счёт полученных значений. Полученные вектора решений ищем в следующей группе таблиц и так далее, пока блоки не закончатся. Вектор, достроенный в последнем блоке будет глобальным вектором решения исходной задачи.

Одним из принципиальных вопросов при исследовании эффективности ЛБЭА является следующий: «В каких случаях применение ЛБЭА в сочетании с некоторым алгоритмом для решения задач ДО в блоках эффективнее использования только лишь упомянутого алгоритма?».

В [81] приведён теоретический анализ оценок эффективности. Ниже описан проведённый сравнительный вычислительный эксперимент, который позволяет оценить эффективность ЛБЭА относительно одного из решателей. Очевидно, что проведение вычислительного эксперимента для ЛБЭА в сочетании со всеми существующими решателями ДО или хотя бы с самыми известными является очень трудоемким процессом. Такого типа эксперименты представлены в [81].

В данной работе для сравнительного эксперимента используются алгоритмы ДО, встроенные в SYMPHONY. SYMPHONY является частью проекта COIN–OR [1] и используется для решения задач частично–целочисленного линейного программирования (ЧЦЛП). Этот решатель выбран из–за открытого исходного кода, переносимости, а также наличия технологии «теплого» старта (ТС), реализующей постоптимальный анализ (ПА) задач ЦЛП [198].

В [199] ПА используется для решения многокритериального дискретного варианта задачи управления инвестициями Марковица с критериями эффективности портфеля и упущенной выгоды. В [200] ПА — для решения задачи о рюкзаке с булевыми переменными. Также ПА широко используется в задачах полубесконечной оптимизации [201] и квадратичных задачах ДО [202].

ПА — это процесс, который реализуется после того, как получено оптимальное решение. Благодаря ПА проявляется чувствительность оптимального решения к определенным изменениям исходной модели. То есть, с помощью ПА анализируется влияние возможных изменений исходных условий на полученное ранее оптимальное решение. Также ПА позволяет подобрать значения параметров в задаче ЦЛП в случае, когда часть параметров задачи не известна и приходится использовать приближенные значения параметров. Кроме того, полученное решение может устареть еще до своей реализации, если не удаётся установить влияние возможных изменений параметров модели на оптимальное решение. Существует графический и аналитический метод ПА.

В ПА исследуются зависимости от следующих параметров:

1. Компоненты вектора ограничений b_t. После нахождения оптимального решения часто необходимо выяснить, как изменение вектора ограничений влияет на оптимальное решение. В качестве примера неравенства модели типа "$\leq$"можно интерпретировать, как ограничения на использование некоторого конечного ресурса, а ограничения типа "$\geq$— как некоторые требования к реализуемому процессу.

2. Коэффициенты ЦФ C_j. Определяются пределы допустимых изменений коэффициентов целевой функции следующим образом:

   — каков диапазон увеличения или уменьшения некоторого коэффициента целевой функции, при оптимальное решение не меняется,

[1] www.coin-or.org

— насколько следует изменить некоторый коэффициент целевой функции, чтобы сделать недефицитный ресурс дефицитным и наоборот.

3. Эффект от изменения коэффициентов ЦФ может рассматриваться со следующих позиций:

   1) необходимо исследовать равновесные компоненты целевой функции (ЦФ);

   2) необходимо исследовать диапазон изменения коэффициента ЦФ, в пределах которого оптимум слабо меняется.

ПА применим для ЛБЭА, поскольку он позволяет:

— использовать при решении задач информацию, полученную при решении уже решенных задач того же пакета;

— эффективно пересчитывать задачу ЦЛП при изменениях параметров задачи.

ТС для реализации ПА используется CBC, Gurobi, SCIP, SYMPHONY и другими решателями. Рассмотрим ТС в SYMPHONY в качестве инструмента для уменьшения перебора при решении подзадач, которые создаются и решаются ЛБЭА. ТС реализован на основе компактного описания дерева поиска в момент приостановки вычислений. Используя данное описание можно сохранять промежуточные вычисления, а затем применять их после перезапуска процесса вычислений в следствие изменения данных задачи.

Далее будут представлены результаты численных расчётов, которые характеризуют эффективность ЛБЭА, то есть время работы этого алгоритма существенно меньше, чем если задача решается напрямую с помощью стандартного метода целочисленного линейного программирования. Целью данного эксперимента является сравнение времени работы трёх алгоритмов для тестовых задач ЦЛП с бинарными переменными, которые имеют БД–структуру, сгенерированную искусственным образом.

Тестовые задачи ЦЛП генерировались исходя из заданного общего количества переменных, связывающих переменных между блоками, а также числа ограничений. Размеры и число блоков вычислялись согласно числу переменных и ограничений исходной задачи. С помощью генератора случайных чи-

Таблица 3.1: Результаты сравнения Symphony без ЛБЭА, Symphony с ЛБЭА и Symphony с ЛБЭА с применением технологии постоптимального анализа

	m	n	k	s	Symphony	Symphony + LEA	Symphony + LEA + PA
1	50	150	25	2	1,5722	0,0301	0,0316
2	100	300	50	4	3,3821	0,1588	0,1579
3	150	500	75	6	—	0,0765	0,0774
4	200	800	100	8	—	0,0367	0,0395
5	250	1000	125	10	—	0,5681	0,5772

сел задавались остальные компоненты задачи: коэффициенты целевой функции, коэффициенты матрицы ограничений и правых частей для каждого блока. Каждая тестовая задача решалась с использованием следующих алгоритмов. Первый алгоритм — это базовый решатель SYMPHONY для задач частично—целочисленного линейного программирования (ЧЦЛП). Второй алгоритм — ЛБЭА, который использовал с решатель SYMPHONY для решения подзадач, соответствующих блокам. Третий алгоритм — вариация второго алгоритма, где решатель SYMPHONY использовался с поддержкой технологии тёплого старта (ТС).

В таблице 3.1 находятся результаты вышеописанного эксперимента. m — число ограничений, n — число переменных задачи, k — число блоков соответствующей структуры, s — максимальный сепаратор, Symphony — время, за которое решается данная задача с помощью Symphony без ЛБЭА, Symphony + LEA — время, за которое решается данная задача с помощью Symphony и ЛБЭА, Symphony + LEA + PA — время, за которое решается данная задача с помощью Symphony и ЛБЭА с использованием постоптимального анализа. Прочерк ставился, если время решения задачи составляет более двух часов.

В результате было установлено явное преимущество второго и третьего алгоритмов над первым. В частности, эксперимент показал, что второй алгоритм становился менее эффективным из–за увеличения объема перебора при решении подзадач в блоках, если увеличивать количество связывающих переменных в задачах с одинаковым числом переменных и размером блоков. В этом случае разумно использовать третий алгоритм, а именно ЛБЭА в сочетании с решателем SYMPHONY, когда используются принципы параметрической оптимизации (технологию ТС). Дело в том, что соответствующие одному и тому же блоку задачи ЦЛП отличаются одна от другой только правыми частями

для разных значений связывающих переменных. Алгоритм получает информацию о решении и использует её для анализа последующих задач, что позволяет решать каждую задачу не полностью, а частично при переборе значений связывающих переменных. Значит с помощью параметрического программирования можно существенно увеличить производительность ЛБЭА. Однако результат эксперимента оказался неоднозначным. Время решения большинства задач с использованием второго и третьего алгоритмов практически сопоставимо. Для некоторых тестовых задач параметрическое программирование оказалась неэффективным. Было замечено также, что что при малых размерностях эффективность ЛБЭА отсутствует.

ЛБЭА достаточно эффективен для разреженных задач согласно [81] и приведённому выше эксперименту, но для достаточно больших размерностей существует серьёзная проблема большого числа вычислений. Далее рассмотрим некоторые модификации, направленные на решение этой проблемы.

3.2 Приближенные методы решений

Приближённые методы широко применяются при решении задач ЦЛП вида (2.1), определённых в первой главе, так как нахождение точного решения может потребовать значительных вычислительных ресурсов. Современные приближенные методы обычно являются комбинированными, т.е. содержат в себе элементы различных методов. Решение задачи приближенными методами обычно происходит в два этапа: построение и улучшение начального решения. При этом на первом этапе широко используются эвристические алгоритмы — алгоритмы, которые не основаны на строго обоснованных предположениях относительно свойств оптимального решения задачи. Примером эвристического алгоритма может быть алгоритм решения задачи коммивояжера, в котором на каждом шаге реализуется переход в ближайшую из оставшихся точку. Алгоритмы такого типа носят название "greedy"(«жадные»алгоритмы).

Эти алгоритмы на каждом шаге решают «локальную» задачу оптимизации; полученное решение может быть далеким от оптимума. Алгоритмы локальной оптимизации, связанные с введенным понятием окрестности, используются на втором этапе; при этом можно использовать несколько алгоритмов этого типа, изменяя правила выбора окрестности.

Локальный элиминационный алгоритм допускает понижение перебора с помощью организации приближенного решения. Мощное направление в развитии приближенных подходов естественным образом возникло внутри точных методов (в основном методов ветвей и границ). При этом неоднократно отмечалось, что для значительного большинства прикладных задач совершенно достаточно вместо точного получить хорошее приближенное решение. Принципиальные вычислительные трудности, возникающие при применении точных методов, и в то же время достаточность хорошего приближенного решения для многих прикладных задач — вот основные источники повышенного интереса к приближенным методам [203].

Трудности с решением достаточно больших подзадач ЛБЭА обусловливают необходимость использования релаксаций. Они позволяют находить и отсеивать решения, которые наверняка не станут оптимальными. Любая задача ЦЛП с ограничениями может быть релаксирована с помощью ослабления ее ограничений. В [204, 205] описывается метод ветвей и границ (МВГ), представляющий собой поиск на дереве с движением от корня дерева поиска к его листьям с использованием оценок для ослабленных задач–релаксаций. В [206] рассматриваются бинарные решающие диаграмм, которые играют роль классических линейных релаксаций для МВГ. В [207] авторы демонстрируют линейную релаксацию для решения квадратичных задач дискретного программирования.

Перейдём к понятию релаксации [205]. Пусть Z — данная задача ЦЛП на максимум. Обозначим через $\mathcal{G}$ множество допустимых решений задачи ЦЛП Z. При этом верхней границей $z_R(x) \geq z(x)$ будем называть такое решение, для которого не выполняются некоторые ограничение, и при этом оптимальное решение меньше. Под релаксацией (Z_R) исходной задачи (Z) будем понимать задачу, для которой выполняются следующие условия:

— $\mathcal{G} \subset \mathcal{G}_R$;

— для $x \in \mathcal{G}$ верхняя граница $z(x) \leq z_R(x)$;

— для $x_1, x_2 \in \mathcal{G}$ из неравенства $z_R(x_1) \leq z_R(x_2)$ следует $z(x_1) \leq z(x_2)$.

Линейная релаксация получается из задачи ЦЛП с помощью отбрасывания условий целочисленности переменных. Релаксация этого вида широко используется и представляет интерес, поскольку полученная задача ЛП может быть

решена с помощью стандартных алгоритмов, например, симплекс–метода либо метода внутренней точки. При реализации линейной релаксация условия $x_j = \{0; 1\}$ заменяются неравенствами $0 \leq x \leq 1$ и в результате получается задача линейного программирования

Ранцевая релаксация состоит в построении из множества ограничений задачи $Z = min\{cx : Ax \geq b, x \in \mathcal{G} \subset R^n, \text{целое}\}$ одного суррогатного ограничения, которое является неотрицательной линейной комбинацией исходных ограничений. При реализации ранцевой релаксации вместо исходной системы ограничений используется так называемое суррогатное ограничение — линейная комбинация исходных ограничений. Таким образом, ранцевая релаксация имеет вид:

$$Z_R(x, u) = min\{cx : uAx \geq ub, x \in \mathcal{G}_R \subset R^n, \text{целое}\}, u \geq 0$$

Релаксированные задачи могут использоваться для зондирования задач ЦЛП. Под прозондированной задачей будем понимать задачу, для которой выполняется одно из следующих условий:

— анализ релаксированной задачи Z_R показал, что Z недопустима. Так, из того, что $\mathcal{G}_R$ — пустое множество, следует, что $\mathcal{G}$ также является пустым множеством;

— анализ релаксированной задачи Z_R показал, что Z не имеет допустимых решений, лучших, чем текущий рекорд. Если $z^{(R)} \leq z_1$ то $z \leq z^{(R)} \leq z_1$;

— анализ релаксированной задачи Z_R позволил найти оптимальное решение Z. Например, если x_R^{opt} — оптимальное решение Z_R, которое допустимо в Z, то x_R^{opt} — оптимальное решение Z.

Основным недостатком ЛБЭА является полный перебор по множествам $S_{rr'}$ сепараторов, что обусловливает возможность большого объема перебора при больших значениях $n_{rr'} = |S_{rr'}|$. Преодолеть этот недостаток и способствовать успешному решению практически важных задач ЦЛП специальной структуры с большими $n_{rr'}$ может введение процедуры, позволяющей «предсказать» искомые оптимальные (или близкие к оптимальным) значения перемычек $x^*_{S_{rr'}}$. Зная значения перемычек $x^*_{S_{rr'}}$, задачи ЦЛП, соответствующие блокам, можно решать отдельно, независимо друг от друга. Очевидно, что возможность узнать

каким–то образом оптимальные значения $x^*_{S_{rr'}}$ представляется сомнительной. Тем не менее можно найти значения перемычек $x_{S_{rr'}}$, близкие к оптимальным, с помощью замены соответствующих блокам задач ЦЛП на их релаксации, а затем решить построенные задачи ЦЛП с помощью ЛБЭА. Таким образом все возможные наборы $x_{S_{rr'}}$ перебираются полностью, однако трудоемкость выполнения каждого шага существенно снижена за счет решения не исходных, а релаксированных и поэтому более легко решаемых подзадач. То есть зная близкие к оптимальным значения перемычек $x_{S_{rr'}}$, можно перебрать некоторую их окрестность с радиусом Хемминга R [2], таким образом, объем перебора составит:

$$P_{rr'} = \sum_{i=0}^{R} C^i_{n_{rr'}}$$

Понятно, что $P_{rr'} = 2^{n_{rr'}}$, радиус R определяется экспериментально, с учетом величины выделяемого для решения задачи времени. Если все же и с релаксированными подзадачами рассматриваемая задача ЦЛП не поддается точному решению, можно использовать случайный выбор значений перемычек $x_{S_{rr'}}$. В данном случае R — число случайным образом порождаемых значений перемычек $x_{S_{rr'}}$. Кроме того, можно использовать неполный перебор перемычек.

Рассмотрим тестирование приближённых модификаций ЛБЭА, которые позволяют существенно ускорить данный алгоритм. Целью вычислительного эксперимента является провести оценку эффективности использования эвристического локального блочно–элиминационного алгоритма (ЭЛБЭА) и ЛБЭА с препроцессингом (ЛБЭАпр), а также сравнить вышеперечисленные приближенные алгоритмы с точным алгоритмом. Все задачи для вычислительного эксперимента являются задачами ЦЛП с булевыми переменными, которые имеют БЛ–структуру. Блоки в каждой задаче имеют одинаковое количество переменных и ограничений, а также одинаковое число переменных в сепараторах между блоками. Длина сепаратора меняется для одних и тех же параметров задачи, поскольку позволяет оценить эффективность каждого из приближенных алгоритмов в зависимости от числа связывающих переменных.

Тестовые задачи генерировались по заданному числу переменных, числу ограничений и размеру сепараторов между блоками. Исходя из количества пе-

[2] Здесь под радиусом Хемминга будем понимать количество компонент, которые отличаются в двух векторах решений.

ременных и ограничений вычислялись размеры блоков и их количество. Далее с помощью датчика случайных чисел генерировались коэффициенты целевой функции, коэффициенты матрицы ограничений и правых частей для каждого из блоков. Библиотека LES (Local Elimination Solver) версии v0.1.1 [3] использовалась для вычисления задач бинарного целочисленного линейного программирования (BILP problem). Матрица ограничений исходной задачи изображена на рисунке 3.1. Здесь k — количество блоков; i_p — количество строк в p–том бло-

$$\begin{array}{|c|}\hline \sum_{j=l_1}^{r_1} a_{1j}x_j \le b_1 \\ \vdots \\ \sum_{j=l_1}^{r_1} a_{i_1 j}x_j \le b_{i_1} \\ \hline \end{array}$$

$$\begin{array}{|c|}\hline \sum_{j=l_2}^{r_2} a_{i_1+1 j}x_j \le b_{i_1+1} \\ \vdots \\ \sum_{j=l_2}^{r_2} a_{i_2 j}x_j \le b_2 \\ \hline \end{array}$$

$$\ddots$$

$$\begin{array}{|c|}\hline \sum_{j=l_k}^{r_k} a_{i_{k-1}+1 j}x_j \le b_{i_{k-1}+1} \\ \vdots \\ \sum_{j=l_k}^{r_k} a_{i_k j}x_j \le b_k \\ \hline \end{array}$$

Рисунок 3.1: Матрица исходной задачи ЦЛП

ке, $p = 1, \ldots, k$; i и j — номера строки и столбца соответственно, $i = 1, \ldots, n$, $j = 1, \ldots, m$; l_p и r_p — левая и правая граница блоков, $l_p < l_{p+1} \le r_p < r_{p+1}$; причем $i_k = n$, $l_1 = 1$ и $r_k = m$. Далее исследуем работу каждого из алгоритмов.

Рассмотрим принцип работы ЭЛБЭА. После того, как построен декомпозиционный граф данной задачи, на его основе строится новый граф, в вершинах которого находятся релаксированные подзадачи. В каждой полученной подзадаче содержится только одно ограничение, определяемое вектором, каждый элемент которого — это сумма элементов первоначальной матрицы ограничений по строкам. Матрица ограничений релаксационной задачи представлена на рисунке 3.2. Значения общих переменных, полученные в результате решения задачи с модифицированным деревом, являются компонентами вектора решений исходной задачи. Затем исходная задача упрощается следующим образом.

[3] https://github.com/robionica/les

$$\sum_{j=l_1}^{r_1} \sum_{i=1}^{i_1} a_{ij} x_j \leq \sum_{i=1}^{i_1} b_i$$

$$\sum_{j=l_2}^{r_2} \sum_{i=i_1+1}^{i_2} a_{ij} x_j \leq \sum_{i=i_1+1}^{i_2} b_i$$

$$\ddots$$

$$\sum_{j=l_k}^{r_k} \sum_{i=i_{k-1}+1}^{i_k} a_{ij} x_j \leq \sum_{i=i_{k-1}+1}^{i_k} b_i$$

Рисунок 3.2: Матрица релаксированной задачи

Значения общих переменных подставляются в задачу так, что изменяются правые части ограничений, а из целевой функции исключаются общие переменные. То есть, в результате данного преобразования получается k не зависящих друг от друга задач, как показано на рисунке 3.3. Здесь x_j^* — значения переменных,

$$\begin{array}{c} \sum_{j=l_1}^{l_2-1} a_{1j} x_j \leq b_1 - \sum_{j=l_2}^{r_1} a_{1j} x_j^* \\ \vdots \\ \sum_{j=l_1}^{l_2-1} a_{i_1 j} x_j \leq b_{i_1} - \sum_{j=l_2}^{r_1} a_{i_1 j} x_j^* \end{array}$$

$$\begin{array}{c} \sum_{j=r_1+1}^{l_3-1} a_{i_1+1 j} x_j \leq b_{i_1+1} - \left(\sum_{j=l_2}^{r_1} + \sum_{j=l_3}^{r_2}\right) a_{i_1+1} x_j^* \\ \vdots \\ \sum_{j=r_1+1}^{l_3-1} a_{i_2 j} x_j \leq b_{i_2} - \left(\sum_{j=l_2}^{r_1} + \sum_{j=l_3}^{r_2}\right) a_{i_2} x_j^* \end{array}$$

$$\ddots$$

$$\begin{array}{c} \sum_{j=r_{k-1}+1}^{r_k} a_{i_{k-1}+1 j} x_j \leq b_{i_{k-1}+1} - \sum_{j=l_k}^{r_{k-1}} a_{i_{k-1}+1 j} x_j^* \\ \vdots \\ \sum_{j=r_{k-1}+1}^{r_k} a_{i_k j} x_j \leq b_{i_k} - \sum_{j=l_k}^{r_{k-1}} a_{i_{k-1}+1 j} x_j^* \end{array}$$

Рисунок 3.3: Матрица упрощенной задачи ЦЛП

полученных после решения релаксированной задачи. Таким образом, вместо исходной задачи решаются релаксированные подзадачи, что позволило существенно сокращает время работы алгоритма.

Результаты вычислительного эксперимента представлены в таблице 3.2. Для проверки эффективности приближенных алгоритмов в качестве тестовых задач были взяты пакеты задач, задачи в которых отличаются только размерами

Таблица 3.2: Результаты работы алгоритма ЭЛБЭА относительно точного алгоритма

	m	n	k	s	Точность	Ускорение	К–во ошибок
1	100	300	10	6	99,31	12,12	3
2	100	300	10	10	98,40	12,52	11
3	100	300	10	14	99,02	4,50	11
4	100	600	10	6	99,85	11,34	1
5	100	600	10	10	99,51	64,26	4
6	100	600	10	14	99,59	3,55	4
7	200	500	10	6	99,49	557,13	4
8	200	500	10	10	99,34	71,37	8
9	200	500	10	14	98,72	3,82	12

сепараторов между блоками. Это позволяет узнать, каким образом увеличение размера сепараторов влияет на точность приближенных алгоритмов.

В таблице 3.2 находятся результаты вышеописанного эксперимента. m — число ограничений, n — число переменных задачи, k — число блоков соответствующей структуры, s — максимальный сепаратор, точность — точность приближенных алгоритмов относительно точного ЛБЭА, ускорение — во сколько раз приближенный алгоритм работает быстрее, чем ЛБЭА, к–во ошибок — сколько переменных из сепараторов было определено неверно.

Рассмотрим тестирование ЛБЭА с препроцессингом (ЛБЭАпр). ЛБЭАпр уменьшает размерность исходной задачи путем исключения переменных и ограничений. Вначале проводится декомпозиция задачи согласно рисунку 3.1. Обозначим блоки задачи — Ω_t, где $t = 1, 2, \ldots, k$ — количество блоков. Зададим вес для Ω_t: определим вес блока как сумму коэффициентов целевой функции при переменных x_j, которые входят в данный блок.

$$w_t = \sum_{j:x_j \in \Omega_t} c_j$$

Разобьем исходную задачу на $3k-2$ следующим образом. Первый и последний блоки разбиваются на две подзадачи, а все остальные — на три: одна из них полностью состоит из локальных переменных данной подзадачи, а две другие — из общих переменных. Правые части ограничений разбиваются в соответствии с весами каждого блока. Рассмотрим произвольный внутренний блок Ω_t, $t = 2, \ldots, k-1$. Пусть L_t — множество локальных переменных, S_{t_l} и S_{t_r} — мно-

жества общих переменных таких, что $S_{t_l} \in \Omega_{t-1} \bigcap \Omega_t$ и $S_{t_r} \in \Omega_t \bigcap \Omega_{t+1}$. Тогда после разбиения на подзадачи блок Ω_t будет выглядеть следующим образом (рис. 3.4):

$$\sum_{j:j\in S_{t_l}} a_{i_b j} x_j \leq b_{i_b 0}$$
$$\sum_{j:j\in S_{t_l}} a_{i_b j} x_j \leq b_{i_b 0}$$

$$\sum_{j:j\in L_t} a_{i_b j} x_j \leq b_{i_b 1}$$
$$\sum_{j:j\in L_t} a_{i_b j} x_j \leq b_{i_b 1}$$

$$\sum_{j:j\in S_{t_r}} a_{i_b j} x_j \leq b_{i_b 2}$$
$$\sum_{j:j\in S_{t_r}} a_{i_b j} x_j \leq b_{i_b 2}$$

Рисунок 3.4: Разбиение блока на подзадачи

Где $i_b = i_{t-1} + 1$ и $i_e = i_t$, $b_{ij} = \frac{b_i}{\sum_{j=0,1,2} b_{ij}}$. Объединим все подзадачи, которые имеют одинаковые множества переменных. После решения этих подзадач исключим из первоначальной задачи все переменные, которые приняли значение 1. Затем соответствующим образом изменим правые части ограничений. Полученное значение целевой функции является искомым.

Так как упрощенная задача может иметь избыточные ограничения, а также переменные, не удовлетворяющие ограничениям, необходимо использовать правила предварительного анализа исходной задачи — препроцессинг. Препроцессинг — фаза между формулированием модели и ее решением, которая применяет простые логические правила (тесты) для переформулировки задачи и сжатия линейной релаксации. При этом препроцессинг может также уменьшить размер задачи в результате фиксирования некоторых переменных и исключения ряда ограничений, а также выявить недопустимость задачи. Простейшее логическое тестирование основано на границах переменных и правых частей ограничений. Пусть L_i — любая нижняя граница, а U_i — любая верхняя граница значения i–й строки $A^i x$ при ограничении $l \leq x \leq u$. Ограничение $A^i x \leq b_i$ избыточно, если $U_i \leq b_i$ и недопустимо, если $L_i > b_i$. Граница переменной может быть уточнена на основе информации о том, что ограничение становится недопустимым при присвоении переменной значения этой границы. Препроцессинг состоит из трех этапов: исключение переменных, не удовлетворяющих хотя бы одному из ограничений, исключение избыточных ограничений (при этом в вектор решений добавляются переменные, которые в любом случае будут принимать значение единицы) и исключение переменных, которые не входят в новые ограничения задачи. Затем решается упрощенная задача с помощью решателя SCIP. Полученные значения переменных записываются в вектор решений, а решение упрощенной задачи добавляется к значению целевой функции.

Таблица 3.3: Результаты, полученные с помощью решателя SCIP и ЛБЭАпр

№	n	m	k	s	Точность	Ускорение
1	40	160	10	3	100,00	0,41
2	40	160	10	5	98,03	0,85
3	40	160	10	7	96,61	1,84
4	80	200	10	3	99,56	0,62
5	80	200	10	5	98,92	0,94
6	80	200	10	7	96,82	3,41
7	120	300	15	3	99,36	1,22
8	120	300	15	5	98,24	1,61
9	120	300	15	7	98,12	6,46
10	100	300	10	3	99,75	2,02
11	100	300	10	5	98,69	2,08
12	100	300	10	7	98,06	1,81
13	150	625	25	3	99,77	0,38
14	150	625	25	5	98,86	4,37
15	150	625	25	7	98,41	76,28
16	100	600	10	3	99,96	1,68
17	100	600	10	5	99,11	3,12
18	100	600	10	7	98,87	7,11
19	200	500	10	3	99,87	4,97
20	200	500	10	5	99,91	37,76
21	200	500	10	7	99,12	157,83

В таблице 3.3 находятся результаты вышеописанного эксперимента. m — число ограничений, n — число переменных задачи, k — число блоков соответствующей структуры, s — максимальный сепаратор, точность — точность приближенного алгоритма относительно точного ЛБЭА, ускорение — во сколько раз приближенный алгоритм работает быстрее, чем ЛБЭА.

Анализ полученных результатов показал, что при достаточно больших сепараторах между блоками квазиблочной задачи ЦЛП приближенные ЛБЭА в сочетании с современными решателями позволяют решать задачи быстрее, чем используемые точные алгоритмы сами по себе при решении всей задачи. Кроме того, с увеличением размерности задач ускорение жадного и эвристического алгоритма становится более сопоставимым. При этом эвристический алгоритм намного чаще «угадывает» значения сепараторов, нежели жадный алгоритм, что позволяет точно решить соответствующую подзадачу. Значит можно предположить, что для задач с многократно большими размерностями более эффективным для использования представляется эвристический локальный элиминационный алгоритм.

3.3 Распараллеливание задач с квазиблочной структурой

В последние годы были представлены результаты работы системы UMPIRE, изложенные у Дж. Форреста и Дж. Томлина [208]. Система UMPIRE предназначена для решения задач ЦЛП различными модификациями метода ветвей и границ. Причем исследовались не только обычные параметры счета (машинное время, число итераций), но и динамика вычислительного процесса (картина построения дерева ветвления). Оказалось, что решавшиеся прикладные задачи принадлежали в основном к одному из двух классов:

I. Задачи, содержащие сравнительно немного (около 100) целочисленных переменных (обычно булевых) и много непрерывных переменных и линейных ограничений сложной структуры (до нескольких тысяч ограничений).

II. Задачи, содержащие значительно больше (до нескольких сотен) целочисленных переменных (в том числе булевых), входящих в ограничения специального вида (такие ограничения возникают обычно из сложных логических условий). «Чисто линейная» часть задачи значительно меньше (до 500 — 1000 ограничений) и проще, чем в I классе задач.

Авторы [208] делают вывод, что задачи промежуточного типа (много целочисленных переменных и сложная и большая «линейная часть» задачи) сравнительно редко встречаются на практике. Представляется, однако, что они должны быть существенно сложнее.

Разработка параллельного программного обеспечения для научных целей — отдельная задача. Выбор целевой вычислительной платформы для реализации ЛБЭАП может существенно влиять на используемый способ распараллеливания, алгоритмические примитивы, используемые для распараллеливания и методы, используемые для отладки параллельного программного обеспечения. Таким образом, развитие параллельного программного обеспечения часто требует большей привязанности к времени и энергии, чем разработки итеративных аналогов. На рисунке 3.5 приведены три типа канонических параллельных вычислительных платформ (ПВП). В [209] показано, как разрабатывать научное параллельное программное обеспечение для каждого из этих трех типов.

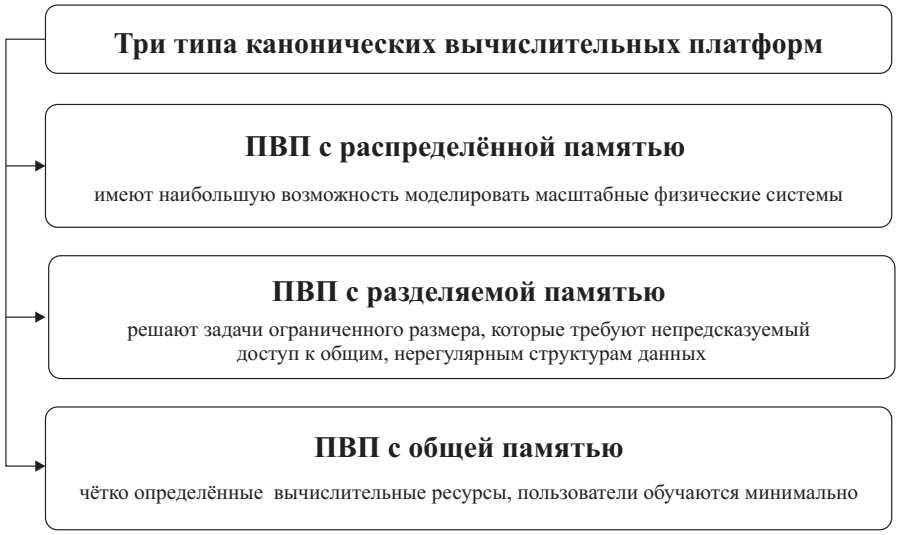


Рисунок 3.5: Три типа канонических вычислительных платформ.

В источниках [210, 211] рассматривается парадигма Master–Worker (MW) или, как в русскоязычной литературе, «Мастер–рабочий» для параллельного метода ветвей и границ (МВГ). Схема MW представляет собой совокупность параллельно работающих процессов, взаимодействующих с помощью сообщений. Управляющий Master–процесс создает первые узлы, а затем распределяет работу между остальными (рабочими) Worker–процессами. Эти рабочие процессы в свою очередь тоже могут состоять из главного и нескольких рабочих процессов. Распределение вычислений управляется пороговыми значениями, с помощью которых задается число ветвлений, которое требуется произвести до

начала обмена данными. Обработка результатов вычислений выполняется в обратном порядке. Преимущества MW: более высокая производительность и понижение загрузки процессора, используя графический процессор (GPU). В статье [212] рассматривается МВГ с поиском в глубину, для параллельной реализации которого выбрана MW парадигма. В [210] спроектирована библиотека BNB–Solver, с помощью которой можно реализовать любой подход, соответствующий схеме MW. В [71] предлагается использование «высокопропускной» вычислительной платформы — кластеры Беовулфа. Такая архитектура однородна (все узлы идентичны), имеет связи «все–во–все», в которой любые два узла могут общаться напрямую, все коммуникации и синхронизация деятельности осуществляются в рамках явного принятия сообщения между процессорами с помощью передачи сообщений протокола MPI.

Для ЛБЭАП могут использоваться системы с распределенной памятью: их параллельное программное обеспечение использует декомпозицию решаемой задачи на подзадачи и назначение их процессорам, а также быструю коммуникационную сеть, обеспечивающую синхронный обмен данными между процессорами. Программирование такой системы может осуществляться с помощью стандартного языка высокого уровня типа C++ или Fortran с явной передачей сообщений. Также может быть использована параллельная система ПО для решения задач смешанного целочисленного линейного программирования (СЦЛП) — SYMPHONY, подобная COIN/BCP [4]. COIN/BCP и SYMPHONY объединены в новом решателе ALPS [213]. В [65] исследуется параллельный решатель SYMPHONY для BCP. Существует целый ряд программных пакетов (рис. 3.6), реализующих параллельные вычисления [71]. В [214] рассматриваются различные параллельные решатели (PUBB, BOB++, PPBB–Lib, PICO, FATCOP, MallBa, ZRAM, BCP, ALPS/BiCePS, MW Framework, Symphony). Кроме того, вопросы ПО для распараллеливания комбинаторных алгоритмов рассматриваются в работах [59, 215–217] [5] [6]. Также на рис. 3.6 показаны решатели, используемые для метода ветвей и границ.

Для решаемых в работе задач выбран язык моделирования AMPL [7]. С помощью AMPLa можно решать линейные и нелинейные задачи оптимизации с

[4] http://www.coin-or.org
[5] http://www.branchandcut.org/
[6] http://www2.cs.uni-paderborn.de/cs/ag-monien/SOFTWARE/PPBB/documentation.html
[7] www.ampl.com

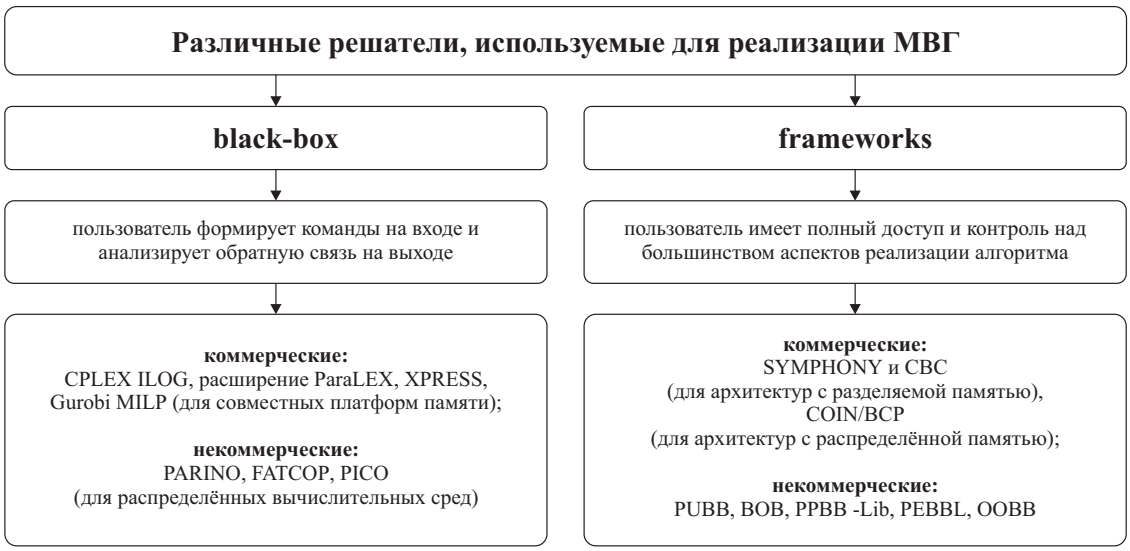


Рисунок 3.6: Различные решатели, используемые для реализации МВГ.

дискретными или непрерывными переменными. При этом AMPL легко встраивается в системы, написанные на других языках программирования и позволяет распараллеливать возникающие подзадачи [218–225].

Ранее при разработке параллельных алгоритмов использовались обычно суперкомпьютеры и кластеры рабочих станций. Рассмотрим подробнее новые возможности применения современных параллельных вычислительных архитектур для ускорения работы ЛБЭАП, такие как графические процессоры (GPU) и GRID.

В последнее десятилетие наблюдается повышенный интерес к GRID–вычислениям [226]. Как указывалось выше, GRID–вычисления используют совокупность слабосвязанных разнородных вычислительных ресурсов как единый вычислительный ресурс. Существенное преимущество этой среды заключается в том, что она предоставляет огромное количество вычислительных ресурсов и позволяет ими пользоваться большому числу пользователей. Создание вычислительных сетей позволяет пользователям использовать простое объединение сотен тысяч подключенных компьютеров и обеспечить необходимую вычислительную мощность. Вычисления GRID представляют собой совокупность распределённых гетерогенных ресурсов, которые могут использоваться в качестве группы выполняемых крупномасштабных приложений. GRID–вычисления используют совокупность слабосвязанных разнородных вычислительных ресурсов как единый объект.

Работа [227] посвящена теоретическому исследованию эффективности решения задачи о ранце в распределённой вычислительной среде и рассмотрен пример реализации МВГ для задач о ранце на GRID системах. Здесь для организации вычислений на GRID выбрана парадигма “Master–workers”. В [228] проведён теоретический анализ и сравнение параллельных МВГ на схемах MW без перераспределения заданий и с перераспределением заданий на примере задачи о рюкзаке. В [229] реализована модель МВГ в системе BNB–GRID и проведён эксперимент на примере задачи нахождения расположения атомов в молекуле с минимальной потенциальной энергией. В [230, 231] демонстрируется, как эффективно решаются задачи комбинаторной оптимизации на GRID. В статье [232] проводилось исследование дерева поиска с помощью распараллеленного МВГ на GRID. Авторы предложили GRID — ориентированный подход, основанный на специальных кодировках исследуемого дерева и рабочих единиц (коллекции узлов). Каждому узлу исследуемого дерева присваивается номер узла. Работа блока ограничена двумя листьями изучаемого дерева, то есть представлена интервалом, начало и конец которого — числа, связанные с двумя листьями. Такой подход позволяет оптимизировать вызванную балансировкой нагрузки взаимосвязь контрольных точек на основе отказоустойчивости и глобальных операций обмена информацией. Обнаружение останова обеспечивает механизм балансировки нагрузки и не требуется никакой дополнительной связи. Другие стратегии балансировки нагрузки рассматриваются в [233–235]. В статье [236] рассмотрена работа МВГ с механизмами отказоустойчивости (МВГО) на GRID. Классические GRID содержат кластеры нескольких физически близких процессоров, подключённых через высокоскоростные сети. МВГО предполагает статическое назначение процессов процессорам так, что ровно один процесс назначается на один физический процессор, поэтому работает быстрее и эффективнее, чем классическая реализация. МВГО полностью распределён и состоит из следующих процедур: начального распределения, балансировки нагрузки, обнаружения завершения, связанного вещания и отказоустойчивости. МВГО считает естественную иерархическую структуру кластера GRID в своих основных процедурах, включая баланс нагрузки и отказоустойчивости, чтобы уменьшить связь в низкоскоростных звеньях. Процессоры и связи довольно часто отказывают, поэтому необходимо работать над высокой степенью отказоустойчивости, иначе расчёты будут ненадёжны. Отказоустой-

чивость позволяет алгоритму восстанавливаться после отдельных неудач процессоров/связей или от неисправности всего кластера процессоров. В МВГО каждый процессор сохраняет таблицы, содержащие информацию о завершённых поддеревьях, и через сообщения распространяются данные этих таблиц. Восстановление после сбоя происходит тогда, когда главный процессор выбирает из своей таблицы незавершённое поддерево и решает его. Одним из методов обнаружения неисправностей является метод, основанный на механизме сердечных сокращений. Этот метод используется, когда из строя выходит целый кластер на GRID. Отказоустойчивость — наиболее важная характеристика МВГО, так как позволяет алгоритму продолжить своё выполнение несмотря на наличие неисправностей. В целях обеспечения надёжности выбирают один из следующих подходов: встроенные механизмы отказоустойчивости в пределах блока программного обеспечения или механизмы отказоустойчивости в пределах алгоритма. Первый подход является более общим, второй приводит к более простым и эффективным процедурам, повышает общую производительность. Ввод/вывод часто является слабым местом параллельных вычислений [209]. Для последовательного ПО процедуру ввода/вывода часто приходится содержать скрытой. При параллельном использовании генераторов псевдослучайных чисел необходимо обеспечить случайные потоки через различные процессоры, так как они являются статистически независимыми, а не коррелируют. Это особенно важно для параллельных вычислений, чьи показатели корректности зависят от независимых случайных величин. Другие приложения, использующие рандомизированные методы моделирования или выборки, включают в себя сортировку и отбор.

В течение последних нескольких лет все более популярной в научном сообществе становится GPU. Технология использования графического процессора GPU, который обычно используется для компьютерной графики с целью выполнения расчётов в приложениях для общих вычислений, которые обычно проводит центральный процессор CPU [237]. Среди программных продуктов, которые позволяют программировать для GPU независимо от платформы, можно упомянуть следующие: CUDA, DirectCompute от Microsoft, OpenGL Shading Language (GLSL). Khronos Group выпустила специализированный язык OpenCL, который является основой для написания программ, выполняемых на разных платформах, состоящих из CPU и GPU. OpenCL является открытым

стандартом, который может быть использован для программирования CPU, GPU и других устройств различных производителей, в то время как CUDA специализирована для NVIDIA GPU. OpenCL обеспечивает переносимость различных аппаратных GPU, ОС, программного обеспечения, а также поддерживается многоядерными процессорами. Поэтому представляется перспективным использование именно OpenCL для реализации алгоритмов структурной декомпозиции разреженных задач ДО.

Существенное ускорение параллельных вычислений можно получить с помощью программируемых графических процессоров (GPU), которые, благодаря своей структуре, могут обрабатывать тысячи потоков данных. Среди публикаций, посвященных параллельным алгоритмам ДО на базе GPU, отметим следующие работы. Обзор [238] параллельного алгоритма муравьиных колоний АМК содержит анализ 106 научных публикаций. Можно отметить, что для параллельной реализации АМК часто используется парадигма «Мастер–рабочий» в основном из-за концептуальной простоты и легкости в применении. В [239, 240] рассмотрены аспекты для параллельной реализации алгоритмов локального поиска и эволюционные алгоритмы на базе GPU. Boyer и соавторы [241] предложили параллельную реализацию метода динамического программирования для задачи о ранце с помощью CUDA для системы из нескольких GPU . Fujimoto и Tsutsui [242] разработали параллельный решатель для задачи о коммивояжере для вычислительной платформы GPU. Gulati и Khatri [243] внедрили новый подход упорядочивания переменных в процедуре решения задачи выполнимости на GPU. Beckers и соавторы [8] предлагают параллельный решатель SAT, реализованный на языке OpenCL на GPU.

Отметим также работы [244] и [245], [246], посвященные вопросам разработки параллельных версий алгоритмов структурной декомпозиции для дискретных задач.

Одним из наиболее популярных алгоритмов для решения задач ДО большой размерности является метод ветвей и границ (МВГ). Для МВГ определены следующие методы распараллеливания [232]: (1) параллельная мульти–параметрическая модель, (2) параллельное исследование дерева, (3) параллельная оценка по времени, и (4) параллельная оценка одного шага (рис. 3.7).

[8] http://hgpu.org/?p=5769

Рисунок 3.7: Методы распараллеливания МВГ [232].

В статье [247] рассматривается реализация фронтального МВГ для параллельных и распределённых систем и исследуется его алгоритмическая сложность. Суть метода заключается в последовательном разбиении конечного множества допустимых точек на подмножества, не содержащие оптимальных решений. Исключение подмножества из дальнейшего рассмотрения производится на основании правила отбора. Каждое подмножество, получаемое в процессе решения, определяет подзадачу оптимизации. Графически процесс работы МВГ можно представить в виде дерева, называемого деревом ветвления. Вершины дерева соответствуют подзадачам, получаемым в результате ветвления, а дуги соединяют данную подзадачу с подзадачами, полученными из неё в результате ветвления. Такой алгоритм называется фронтальным параллельным МВГ. В [248] рассматриваются основные принципы построения параллельных алгоритмов, а также параллельные алгоритмы МВГ для решения задач комбинаторной и целочисленной оптимизации и параллельные алгоритмы ВС. В статье [66] рассматривается параллельная реализация ВСР. В [249] предлагается увеличить число альтернативных деревьев поиска (лес поиска), чтобы быстро найти лучшее решение. Авторы назвали этот способ multiple heuristic search (MHS) — мульти–эвристическим поиском. В этом случае лучшее целое решение транслируется в процессе исследования дерева с помощью MHS. В [250] выделены несколько типов распараллеливания МВГ: parallelizing of operations (PO) — распараллеливание операций, parallelizing of decomposition (PD) — распараллеливание декомпозиции, parallelizing of algorithm copies (PAC) — распараллеливание копий алгоритма (рис. 3.8).

В статье [251] предлагается использовать вышеперечисленные методы распараллеливания для задачи определения местоположения депо и управления

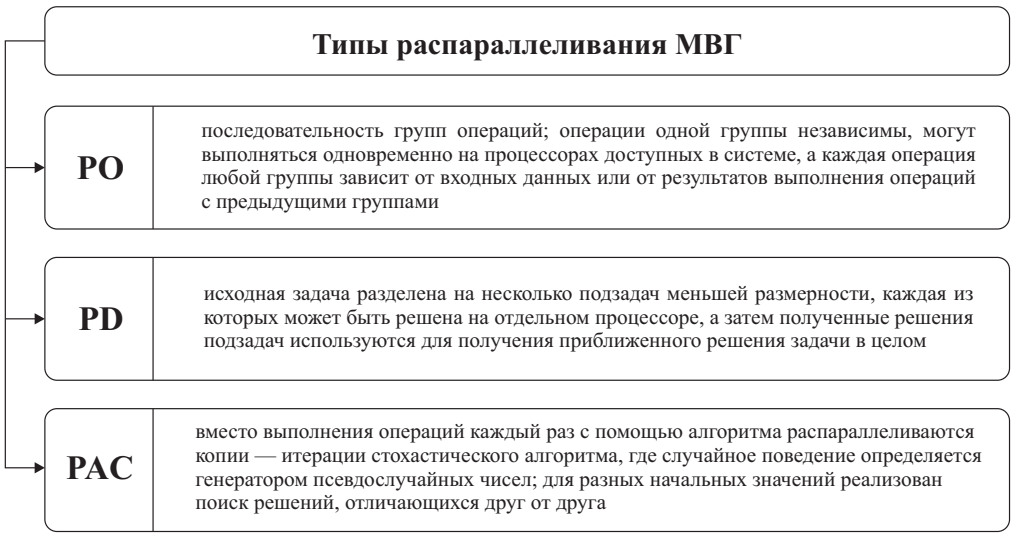


Рисунок 3.8: Типы распараллеливания МВГ [250].

контейнерного парка. Представленные подходы распараллеливания используются для МВГ, где параллелизм получается путём деления дерева поиска на нескольких подзадач одновременно между процессорами и выполнением операций. Авторы заметили, что эффективность последовательного и параллельного МВГ для небольших деревьев одинакова, но её характеристики начинают ухудшаться, если размер дерева растет и стратегии, реализующие коллегиальный контроль списка подзадач, работают лучше. Так же в статье описываются результаты работы гибридной схемы сразу нескольких подходов.

В [252] описывается распараллеливание МВГ. Последовательный МВГ заключается в неявном переборе подмножеств допустимых решений. Перебор решений задачи состоит в создании дерева поиска МВГ, вершины которого являются множествами решений рассматриваемой задачи. Размер дерева, то есть количество создаваемых вершин, непосредственно связан с методом, используемым для его построения. В настоящее время известны две основные стратегии распараллеливания для МВГ:

1. Вершинные стратегии, цель которых — ускорить конкретную операцию на уровне вершины: параллельное вычисление нижней или верхней границы, параллельная оценка вершин–потомков, и т.д.

2. Древовидные стратегии, цель которых — параллельное построение и исследование дерева поиска ветвей и границ.

Вычислительная сложность решения задачи с помощью МВГ определяется как число выполненных шагов метода. Определённая сложность совпадает с

числом подвергнутых ветвлений или подзадач, исключённых из рассмотрения по правилам отсева. В статье [253] рассматривается параллельная реализация МВГ для решения задачи о ранце и оценивается его вычислительная сложность. В статье [71] рассмотрены проблемы (рис. 3.9), которые возникают при распараллеливании МВГ для решения задач смешанного целочисленного линейного программирования (MILPs) и спроектировано программное обеспечение для решения этих проблем.

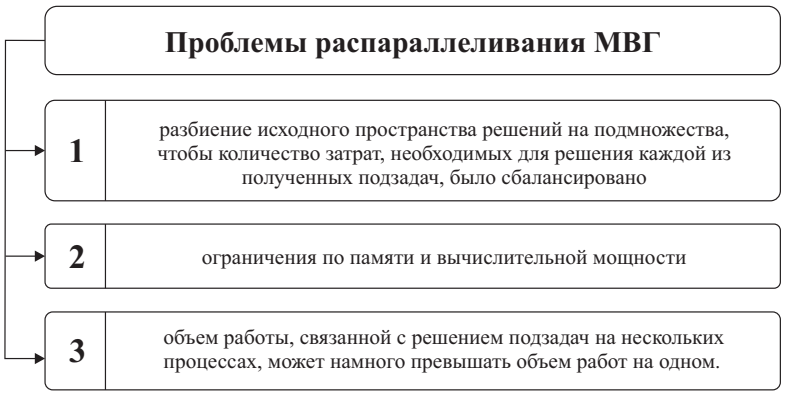


Рисунок 3.9: Проблемы распараллеливания МВГ

При распараллеливании МВГ можно отметить следующие сложности: структура дерева и граф зависимостей между задачами заранее неизвестны, задачи для процессоров создаются динамически в процессе работы алгоритма. Для того, чтобы избежать дополнительных временных затрат, необходимо принимать во внимание такие алгоритмические аспекты, как балансировка нагрузки и передача сообщений между процессорами.

Как показано в работе [83], для достаточно разветвлённого ЭД имеет смысл применять технологию распараллеливания, так как подзадачи, находящиеся на одной высоте дерева, являются независимыми. Такой вид распараллеливания будем называть древовидным распараллеливанием. Рассмотрим подзадачу, соответствующую вершине ЭД. Она имеет блочную структуру, так как первичная задача была разреженной. Блоки такой подзадачи слабо связные, поэтому имеет смысл разбивать такие блоки, перебирая переменные, которые являются сепараторами этих блоков. Такой вид распараллеливания будем называть блочным. ЛБЭА дважды проходит по ЭД. Прямой ход ЛБЭА решает подзадачи и сохраняет промежуточные решения, а обратный ход ЛБЭА анализирует и собирает решения подзадач.

Далее рассмотрим эксперимент, реализующий распараллеливание БЛ– и БД–задач на GRID. Целью данного эксперимента является реализация решение задач с квазиблочной структурой с помощью параллельной модификации ЛБЭА — ЛБЭАП. Для параллельной реализации ЛБЭА использовалась облачная платформа Everest, поддерживающая публикацию, выполнение и композицию вычислительных приложений в распределенной среде [254, 255]. Одной из отличительных черт платформы является возможность запуска приложений на произвольных комбинациях внешних вычислительных ресурсов, подключенных пользователями.

Программирование ЛБЭА для платформы Everest осуществлялось с помощью языка алгебраического программирования AMPL. Так как AMPL не решает задачи непосредственно, в качестве соответствующего внешнего «решателя» использовался пакет SCIP — широко используемая система с открытым кодом [9]. При этом транслятор AMPL использовался не только для формулировки основной задачи и всех вспомогательных подзадач, но и для управления сценарием расчетов в распределенной среде солверов, подключенных к Everest, с помощью подсистемы AMPLX [256], доступной в исходных кодах [10].

Для исследования ЛБЭА были сгенерированы разреженные задачи ЦЛП, в которых можно выделить БД–структуру. Задачи с БД–структурой содержали 50 000 переменных и 100 ограничений и решались в среднем за 17 минут (рис А.1). Задачи разбивались на 15 подзадач с глубиной дерева — 5. При этом подзадачи могли содержать максимально 10 000 переменных 20 ограничений, размер сепаратора — 4, а число вершин потомков — 12.

Рассмотрим пошагово решение задачи с БД–структурой согласно ЛБЭА. На рисунке 3.10 изображена структура задачи, внутри блоков указано число независимых переменных в каждом блоке. Слева указан номер блока, справа — число ограничений в данном блоке. Над связью между блоками указано число переменных, которые одновременно присутствуют в этих блоках.

Рассмотрим работу ЛБЭА на примере большой задачи предложенной БД–структурой. С помощью модифицированного алгоритма Финкельштейна был определён порядок решения подзадач: $(\{15, 14, 13, 12\}; \{11, 10, 9\}; \{8, 7, 6, 5\}; \{4, 3, 2\}; \{1\})$, где в фигурных скобках выделены блоки, которые могут быть решены параллельно.

[9] http://scip.zib.de/

[10] https://gitlab.com/ssmir/amplx

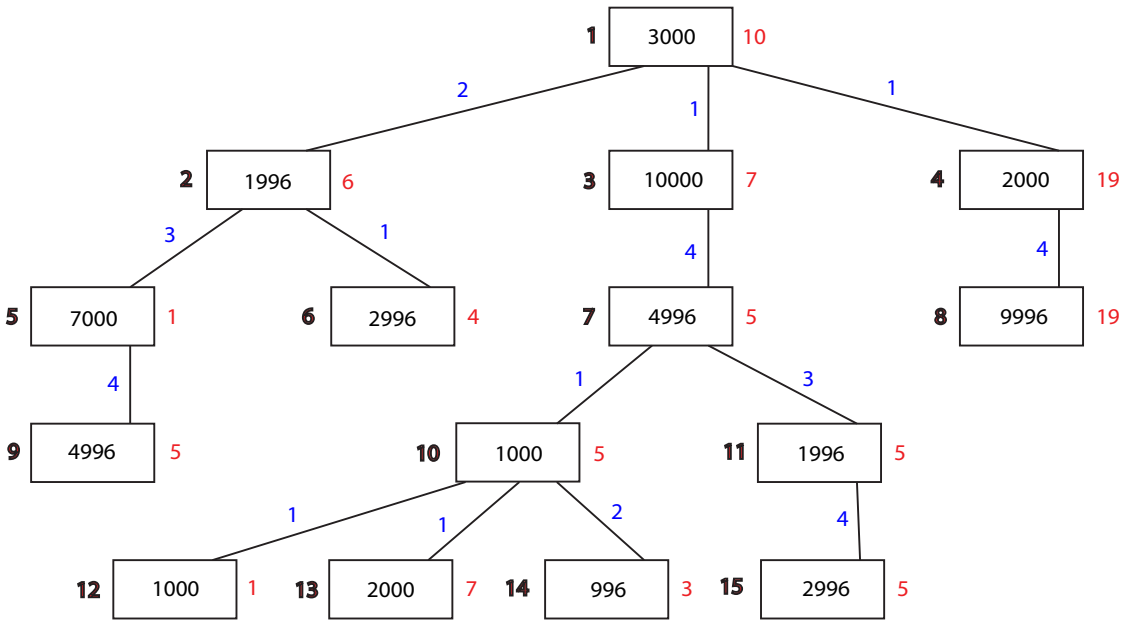


Рисунок 3.10: Задача с БД–структурой

На первой итерации решаем параллельно подзадачи, соответствующие блокам ($\{15, 14, 13, 12\}$. У этих блоков нет потомков, поэтому они содержат только два типа переменных: блочные переменные — $\{35005, \ldots, 38000\}$; $\{39005, \ldots, 40000\}$; $\{40001, \ldots, 42000\}$; $\{42001, \ldots, 43000\}$ и сепараторы — $\{37001, \ldots, 37004\}$; $\{39003, 39004\}$; $\{39002\}$; $\{39001\}$. Создадим подзадачи, соответствующие данным блокам: $P_{15}(\{37001, \ldots, 37004\})$, $P_{14}(\{39003, 39004\})$, $P_{13}(\{39002\})$, $P_{12}(\{39001\})$ и зафиксируем сепараторы каждого из этих блоков $X^s_{15} = X^s_{14} = X^s_{13} = X^s_{12} = \{0, 0, \ldots, 0, 0\}$. Решим 4 полученные подзадачи и занесём решения $(X^s_{15}, X^t_{15}, Z_{15})$; $(X^s_{14}, X^t_{14}, Z_{14})$; $(X^s_{13}, X^t_{13}, Z_{13})$; $(X^s_{12}, X^t_{12}, Z_{12})$ в таблицы ЛБЭА, где X^s_{15}, X^s_{14}, X^s_{13}, X^s_{12} — зафиксированное значение сепараторов, X^t_{15}, X^t_{14}, X^t_{13}, X^t_{12} — значение векторов свободных переменных и Z^s_{15}, Z^s_{14}, Z^s_{13}, Z^s_{12} — экстремальные значения целевых функций. По аналогии решим остальные подзадачи для каждого значения сепаратора от $X^s_{15} = X^s_{14} = X^s_{13} = X^s_{12} = \{0, 0, \ldots, 0, 1\}$ до $X^s_{15} = X^s_{14} = X^s_{13} = X^s_{12} = \{1, 1, \ldots, 1, 1\}$, всего получится 2^{s_r} подзадач для каждого r–того блока, где $r = 12, \ldots, 15$.

Далее рассмотрим следующую группу блоков $\{11, 10, 9\}$, соответствующие подзадачи которых также решаются параллельно. У 11 и 10 блоков есть потомки, а у 9–го потомков нет. Сначала рассмотрим блок 9: у этого блока нет потомков, поэтому он содержат только два типа переменных: блочные переменные — $\{15005, \ldots, 20000\}$ и сепараторы — $\{15001, \ldots, 15004\}$. Создадим подзадачу $P_9(\{15001, \ldots, 15004\})$ и зафиксируем сепараторы $X^s_9 = \{0, 0, \ldots, 0, 0\}$.

Решим полученную подзадачу и занесём решение (X_9^s, X_9^t, Z_9) в таблицу ЛБЭА, где X_9^s — зафиксированное значение сепараторов, X_9^t — значение вектора свободных переменных и Z_9^s — экстремальное значение целевой функции. По аналогии решим остальные подзадачи для каждого значения сепаратора от $X_9^s = \{0, 0, \ldots, 0, 1\}$ до $X_9^s = \{1, 1, \ldots, 1, 1\}$, всего получится 2^{s_9} подзадач.

Далее рассмотрим блоки 11 и 10: у этих блоков есть потомки, поэтому соответствующие подзадачи строятся следующим образом. Блоки 11 и 10 содержат три типа переменных: блочные переменные — $\{35005, \ldots, 37000\}$; $\{38001, \ldots, 39000\}$, сепараторы — $\{35002, \ldots, 35004\}$; $\{35001\}$ и наследственные переменные — $\{37001, \ldots, 37004\}$; $\{39001, \ldots, 39004\}$. Множество наследственных переменных $\{37001, \ldots, 37004; 39001, \ldots, 39004\}$ разбиваются на четыре подмножества: $\{37001, \ldots, 37004\}$; $\{39003, 39004\}$; $\{39001\}$; $\{39002\}$, так как у блоков 10 и 11 число потомков равно 4. Создадим подзадачи $P_{11}(X_{11}^s)$ и $P_{10}(X_{10}^s)$ и зафиксируем сепараторы этих блоков $X_{11}^s = X_{10}^s = \{0, 0, \ldots, 0, 0\}$. Наследственные переменные для соответствующих подзадач представлены в виде четырёх табличных функций, соответствующих блокам–потомкам $\{15; 14; 13; 12\}$: согласно значениям переменных блоков–потомков из каждой соответствующей таблицы выбираются значения целевой функции Z_{15}, Z_{14}, Z_{13}, Z_{12} и прибавляется к значению соответствующей целевой функции. Таким образом целевая функция для 11 блока — $Z_{11} + Z_{15}$, а для 10 блока — $Z_{10} + Z_{12} + Z_{13} + Z_{14}$. Решения задачи для всех значений сепаратора заносятся в соответствующие таблицы. Всего таких подзадач для блоков 10 и 11 $2^{s_{11}} \cdot 2^{s_{15}} + 2^{s_{10}} \cdot 2^{s_{14}+s_{13}+s_{12}}$. То есть на этом этапе распараллеливания получается $2^{s_{11}} \cdot 2^{s_{15}} + 2^{s_{10}} \cdot 2^{s_{14}+s_{13}+s_{12}} + 2^{s_9}$ подзадач.

Далее рассмотрим следующую группу блоков $\{8; 7; 6; 5\}$, соответствующие подзадачи которых также решаются параллельно. У 7 и 5 блоков есть потомки, а у блоков 8 и 6 потомков нет. Сначала рассмотрим блоки 8 и 6: у этих блоков нет потомков, поэтому они содержат только два типа переменных: блочные переменные — $\{45005, \ldots, 50000\}$; $\{5005, \ldots, 8000\}$ и сепараторы — $\{45001, \ldots, 45004\}$; $\{5004\}$. Создадим подзадачи $P_8(\{45001, \ldots, 45004\})$ и $P_6(\{5004\})$ и зафиксируем сепараторы каждого из этих блоков $X_8^s = X_6^s = \{0, 0, \ldots, 0, 0\}$. Решим полученные подзадачи и занесём решения (X_8^s, X_8^t, Z_8) и (X_6^s, X_6^t, Z_6) в таблицы ЛБЭА, где X_8^s и X_6^s — зафиксированное значение сепараторов, X_8^t и X_6^t — значение векторов свободных переменных и Z_8^s и Z_6^s

— экстремальные значения целевых функций. По аналогии решим остальные подзадачи для каждого значения сепараторов от $X_8^s = X_6^s = \{0, 0, \ldots, 0, 1\}$ до $X_8^s = X_6^s = \{1, 1, \ldots, 1, 1\}$, всего получится $2^{s_8} + 2^{s_6}$ подзадач.

Далее рассмотрим блоки 7 и 5: у этих блоков есть потомки, поэтому соответствующие подзадачи строятся следующим образом. Блоки 7 и 5 содержат три типа переменных: блочные переменные — $\{30005, \ldots, 35000\}$; $\{8001, \ldots, 15000\}$, сепараторы — $\{30001, \ldots, 30004\}$; $\{5001, \ldots, 5003\}$ и наследственные переменные — $\{35001, \ldots, 35004\}$; $\{15001, \ldots, 15004\}$. Так как у блоков 7 и 5 число потомков равно 3, множество наследственных переменных $\{35001, \ldots, 35004;$ $15001, \ldots, 15004\}$ разбиваются на 3 подмножества: $\{15001, \ldots, 15004\}$; $\{35001\}$; $\{35002, \ldots, 35004\}$. Создадим подзадачи $P_7(X_7^s)$ и $P_5(X_5^s)$ и зафиксируем сепараторы этих блоков $X_7^s = X_5^s = \{0, 0, \ldots, 0, 0\}$. Наследственные переменные для соответствующих подзадач представлены в виде четырёх табличных функций, соответствующих блокам–потомкам $\{11; 10; 9\}$: согласно значениям переменных блоков–потомков из каждой соответствующей таблицы выбираются значения целевой функции Z_{11}, Z_{10}, Z_9 и прибавляется к значению соответствующей целевой функции. Таким образом целевая функция для 7 блока — $Z_7 + Z_{10} + Z_{11}$, а для 5 блока — $Z_5 + Z_9$. Решения задачи для всех значений сепаратора заносятся в соответствующие таблицы. Всего таких подзадач для блоков 7 и 5 — $2^{s_7} \cdot 2^{s_{11}+s_{10}} + 2^{s_5} \cdot 2^{s_9}$. То есть на этом этапе распараллеливания получается $2^{s_7} \cdot 2^{s_{11}+s_{10}} + 2^{s_5} \cdot 2^{s_9} + 2^{s_8} + 2^{s_6}$ подзадач.

Далее рассмотрим следующую группу блоков $\{4, 3, 2\}$, соответствующие подзадачи которых также решаются параллельно. У всех этих блоков есть потомки, поэтому соответствующие подзадачи строятся следующим образом. Блоки 4, 3 и 2 содержат три типа переменных: блочные переменные — $\{43001, \ldots, 45000\}$; $\{20001, \ldots, 30000\}$; $\{3005, \ldots, 5000\}$, сепараторы — $\{3004\}$; $\{3003\}$; $\{3001, 3002\}$ и наследственные переменные — $\{45001, \ldots, 45004\}$; $\{30001, \ldots, 30004\}$; $\{5001, \ldots, 5004\}$. Так как число блоков–потомков равно четырём, множество наследственных переменных $\{45001, \ldots, 45004; 30001, \ldots, 30004; 5001, \ldots, 5004\}$ разбиваются на четыре подмножества: $\{5001, \ldots, 5003\}$; $\{5004\}$; $\{30001, \ldots, 30004\}$; $\{45001, \ldots, 45004\}$. Создадим подзадачи $P_4(X_4^s)$, $P_3(X_3^s)$ и $P_2(X_2^s)$ и зафиксируем сепараторы этих блоков $X_4^s = X_3^s = X_2^s = \{0, 0, \ldots, 0, 0\}$. Наследственные переменные для соответствующих подзадач представлены в виде четырёх табличных функций, со-

ответствующих блокам–потомкам $\{8; 7; 6; 5\}$: согласно значениям переменных блоков–потомков из каждой соответствующей таблицы выбираются значения целевой функции Z_8, Z_7, Z_6, Z_5 и прибавляется к значению соответствующей целевой функции. Таким образом целевая функция для 4 блока — $Z_4 + Z_8$, для 3 блока — $Z_3 + Z_7$, а для 2 блока — $Z_2 + Z_6 + Z_5$. Решения задачи для всех значений сепаратора заносятся в соответствующие таблицы. Всего таких подзадач для блоков 4, 3 и 2 — $2^{s_4} \cdot 2^{s_8} + 2^{s_3} \cdot 2^{s_7} + 2^{s_2} \cdot 2^{s_6+s_5}$.

Рассмотрим последний блок, у него есть потомки, но нет сепараторов, так что он содержит два типа переменных: блочные переменные — $\{1, \ldots, 3000\}$ и наследственные переменные — $\{3001, \ldots, 3004\}$. Так как число блоков–потомков равно 3, множество наследственных переменных $\{3001, \ldots, 3004\}$ разбиваются на 3 подмножества: $\{3001, 3002\}$; $\{3003\}$; $\{3004\}$. Создадим подзадачу P_1 и найдём её решение. Наследственные переменные для последней подзадачи представлены в виде трёх табличных функций, соответствующих блокам–потомкам $\{4; 3; 2\}$: согласно значениям переменных блоков–потомков из каждой соответствующей таблицы выбираются значения целевой функции Z_4, Z_3, Z_2 и прибавляется к значению соответствующей целевой функции. Таким образом целевая функция для последнего блока — $Z_1 + Z_2 + Z_3 + Z_4$. Решение текущей подзадачи является глобальным решением задачи. На этом шаге параллельно решаются $2^0 \cdot 2^{s_4+s_3+s_2}$ подзадач.

Теперь необходимо восстановить глобальное решение. Для восстановления глобального решения на каждом шаге необходимо искать значение векторов решений в таблицах блоков–потомков 4, 3 и 2 последнего блока среди значений сепараторов $\{3001, 3002\}$; $\{3003\}$; $\{3004\}$. Соответствующие строки таблиц будут содержать решения подзадач, причём если в одном блоке имеется несколько решений, выбираем оптимальное из них и расширяем вектор решений $\{1, \ldots, 3004\}$ за счёт полученных значений. Теперь нам известны значения переменных $\{1, \ldots, 5000;\ 20001, \ldots, 30000;\ 43001, \ldots, 45000\}$. Полученные вектора решений ищем в следующей группе таблиц, соответствующих блокам $5, \ldots, 8$ среди значений сепараторов $\{5001, \ldots, 5004\}$; $\{30001, \ldots, 30004\}$; $\{45001, \ldots, 45004\}$. На этом шаге нам известны значения следующих групп переменных: $\{1, \ldots, 15000;\ 20001, \ldots, 35000;\ 43001, \ldots, 50000\}$. Рассмотрим следующий ряд таблиц, соответствующую блокам 9, 10 и 11. Среди значений сепараторов $\{15001, \ldots, 15004\}$; $\{35001\}$; $\{35002, \ldots, 35004\}$ ищем зна-

чения блочных переменных. Теперь известны $\{1,\ldots,37000;\ 38001,\ldots,39000;\ 43001,\ldots,50000\}$. Наконец рассмотрим последнюю группу таблиц, соответствующую блокам $12,\ldots,15$. Ищем строки согласно значениям сепараторов $\{39001\};\{39002\};\{39003,39004\};\{37001,\ldots,37004\}$. Достроенный вектор является глобальным вектором решения исходной задачи.

Задачи с БЛ–структурой содержали 100 000 переменных и 100 ограничений и решались в среднем за 6 часов (рис А.2). Задачи разбивались на 5 подзадач с глубиной дерева — 5. При этом подзадачи могли содержать максимально 20 000 переменных 20 ограничений, размер сепаратора и число вершин потомков — 4.

Предложенные задачи с БД–структурой и БЛ–структурой не могут быть решены точно только с помощью «решателя» SCIP без ЛБЭА.

На рисунках А.1 и А.2 (приложение А) изображёны профили (интервалы времени выполнения подзадач на подключенных ресурсах, всего — 16) вычислительных процессов для задач, имеющих БД– и БЛ–структуру. На рисунке А.1 виден хороший баланс подзадач, решение исходной задачи происходит за 17 минут. На рисунке А.2 виден плохой баланс подзадач, решение исходной задачи происходит за 335 минут. Здесь большое число подзадач обрабатывается очень быстро. Причина в том, что большое число подзадач несовместны. К сожалению, препроцессинг AMPLa не позволяет регулярным образом «пропустить» заведомо несовместную подзадачу. Поэтому препроцессинг приходится отключать и «поручать» проверку несовместности решателям.

Основным результатом данного эксперимента является реализация ЛБЭАП для решения задач ЦЛП со специальной структурой с помощью GRID. В результате проведенного вычислительного эксперимента было отмечено, что для задач с БД– и БЛ–структурой ЛБЭАП гораздо эффективнее, чем решатель без ЛБЭАП.

Заключение

Основные результаты работы заключаются в следующем. Разработана техника понижения размерности разреженных матриц и соответствующих задач ДО большой размерности за счёт выделения квазиблочных структур и последующей их обработки.

В данной работе удалось сформировать метод выделения квазиблочных структур для разреженных матриц, разработать и реализовать алгоритмы выделения таких структур и последующего решения соответствующих задач ДО. Осуществлено распараллеливание задач ДО на GRID. Это позволило получить решение для задач, которые в силу размерности не могут быть решены с помощью обычных компьютеров.

Поставлен ряд новых задач в рассматриваемой области, в частности, оценка количества вычислений при распараллеливании, выделение классов задач с полиномиальной сложностью при выборе порядка исключений в локальном алгоритме, применение теорем о составе квазиблочной структуры для оптимального выбора количества процессоров при распараллеливании, а также оценка количества вычислений для алгоритмов выделения блочно–лестничной и блочно–древовидной структуры.

Литература

1. Cook S. A. Feasibly constructive proofs and the propositional calculus // Proc. ACM. 1975. P. 83–97.

2. Cook S., Nguyen P. Logical Foundations of Proof Complexity. New York: Cambridge University Press, 2014. 496 p.

3. Авербах И. Л., Цурков В. И. Оптимизация в больших задачах с целочисленными переменными. М.: Наука, Физматлит, 1995. 288 с.

4. Авербах И. Л. К одному методу декомпозиции в блочном целочисленном программировании // Декомпозиция и координация в сложных системах. Ч. 1. Челябинск: ЧПИ, 1986. С. 50–51.

5. Авербах И. Л. Теоретико–групповой метод декомпозиции в целочисленном линейном программировании // Ж. вычисл. матем. и матем. физ. 1992. Т. 8, № 32. С. 1229–1243.

6. Афонин Ю. С. Блочный метод ветвей и границ // Автоматика и телемеханика. 1986. № 8. С. 98–108.

7. Аюпов Р. С. Комбинаторно–непрерывный метод решения блочных задач дискретного программирования // Изв. АН СССР. Техн. кибернетика. 1984. № 3. С. 34–39.

8. Бахтин А. Е. Блочный способ решения задач линейного программирования // Математические методы решения экономических задач. 1971. С. 63–78.

9. Бахтин А. Е., Колоколов А. А. Декомпозиционный метод решения целочисленных производственно–транспортных задач // Вопросы анализа сложных систем. 1974. С. 15–36.

10. Верина Л. Ф., Танаев В. С. Декомпозиционные подходы к решению задач математического программирования // Экономика и математические методы. 1976. № 6. С. 1160–1172.

11. Верина Л. Ф. О декомпозиции задач линейного программирования квазиблочной структуры // Весці АН БССР. Сер. физ.-мат. наук. 1975. № 6. С. 18–21.

12. Гольденгорин Б. И. Декомпозиция задачи размещения // Автоматика и телемеханика. 1986. № 5. С. 91–101.

13. Гольденгорин Б. И. Алгоритм декомпозиции задачи унификации и новые полиномиально разрешимые случаи // Докл. АН СССР. Т. 288. 1986. С. 19–23.

14. Емеличев В. А., Тыонг Буй Кат. Декомпозиционный подход к решению квазиблочных задач дискретной оптимизации на основе метода построения последовательности планов // Кибернетика. 1988. № 1. С. 116–118.

15. Канцедал С. А. Декомпозиционный подход к решению задач теории расписаний большой размерности // Автоматика и телемеханика. 1983. № 10. С. 57–58.

16. Ковалёв М. М. Декомпозиционный подход к построению E-оптимальных приближенных алгоритмов // Математическое и прогр. обеспечение вычислит. систем. 1985. С. 3–8.

17. Кравец В. Л., Сергиенко И. В. Декомпозиционный метод решения одного класса комбинаторных оптимизационных задач // Кибернетика. 1983. № 6. С. 77–84.

18. Кутиков Л. М. Декомпозиция блочных задач линейного программирования со слабо связанными блоками // Экономика и матем. методы. 1973. Т. 4, № 9. С. 739–749.

19. Левин Г. М., Танаев В. С. Параметрическая декомпозиция экстремальных задач // Изв. АН БССР. 1974. № 4. С. 24–29.

20. Левин Г. М., Танаев В. С. Декомпозиционные методы оптимизации проектных решений. Минск: Наука и техника, 1978. 240 с.

21. Малинников В. В. Метод разложения в решении задач линейного программирования с блочной структурой // Экономика и матем. методы. 1971. Т. 5, № 7. С. 733–736.

22. Мащенко С. О. Декомпозиционный алгоритм решения частично целочисленных задач линейного программирования // Исследование задач многокритериальной оптимизации. 1984. С. 49–63.

23. Павлов А. А. Об одном методе декомпозиции в задачах программного управления динамическими системами с постоянной и переменной структурами автоматического управления // Адаптивные системы автоматического управления. 1985. № 13. С. 37–40.

24. Росс Г. В. Декомпозиционный алгоритм решения задачи смешанного целочисленного программирования для составления расписания // Алгоритм. обеспеч. и прооектир. микропроцессорных управл. систем. 1982. С. 25–30.

25. Рощин В. А., Семёнова Н. В., Сергиенко И. В. Декомпозиционный подход к решению некоторых задач целочисленного программирования с неточными данными // Журн. вычисл. математики и матем. физики. 1990. Т. 5, № 29. С. 786–791.

26. Сергиенко И. В., Голодников А. Н. О применении метода вектора спада в декомпозиционных схемах решения задач целочисленного линейного программирования // Кибернетика. 1984. С. 44–47.

27. Танаев В. С. Декомпозиция и агрегирование в задачах математического программирования. Минск: Наука и техника, 1987. 183 с.

28. Тесленко Е. А. Решение задач большой размерности методом декомпозиции // Моделиров. и информат. в РАС и ОГАС. 1983. № 6. С. 86–94.

29. Уздемир А. П. Схема последовательной декомпозиции в задачах оптимизации // Автоматика и телемеханика. 1980. № 11. С. 94 –105.

30. Цурков В. И. Декомпозиция в задачах большой размерности. М.: Наука, 1981. 352 с.

31. Avramita G. M. Decomposition and solving of large–scale problems in production scheduling // Economic Computation and Economic Cybernetics Studies and Research. 1978. Vol. 12, no. 1. P. 63–76.

32. Burkard R. E., Hamacher H. W., Tind J. On general decomposition schemes in mathematical programming // Math. Program. Study. 1985. Vol. 24. P. 238–252.

33. Giulianelly S., Lucertini M. A decomposition technique in integer linear programming // Lecture Notes Comput. Sci. 1976. Vol. 41. P. 86–97.

34. Magnati T. L., Wong R. T. Accelerating Benders decomposition: algorithmic enhancement and model selection criteria // Operations Research. 1981. Vol. 29, no. 3. P. 464–484.

35. Nowak I. Lagrangian decomposition of block–separable mixed–integer all–quadratic programs // Math. Programming. 2005. Vol. 102, no. 2. P. 295–312.

36. Ralphs T. K., Galati M. Decomposition in Integer Programming / Ed. by J. Karlof. 2005. p. 57.

37. Sannomiya N., Okamoto K. A method for decomposing mixed-integer linear programs with staircase structure // Int. J. Syst. Sci. 1985. Vol. 16, no. 1. P. 99–111.

38. Schrage L. Using decomposition in integer programming // Naval Research Logistics Quarterly. 1973. Vol. 20, no. 3. P. 469–476.

39. Лэсдон Л. С. Оптимизация больших систем. М.: Наука, 1975. 432 с.

40. Crowder H., Johnson E. L., Padberg M. W. Solving large–scale zero–one linear programming problems // Operations Research. 1983. Vol. 31. P. 803–834.

41. Ф. А. Мурзин С. А. Полетаев. История развития суперкомпьютерной вычислительной техники // Конструирование и оптимизация параллельных программ. 2008. С. 174–215.

42. Полетаев С. А. Параллельные вычисления на графических процессорах // Конструирование и оптимизация параллельных программ. 2008. С. 270–300.

43. Воеводин В. В. Математические модели и методы в параллельных процессах. М.: Наука, 1986. 296 с.

44. Глушков В. М., Молчанов И. Н. О некоторых проблемах решения задач на ЭВМ с параллельной организацией вычислений // Кибернетика. 1981. № 4. С. 82–88.

45. Евреинов Э. Р., Косарев Ю. Г. О решении задач на универсальных системах // Вычисл. системы. 1965. № 17. С. 106–164.

46. Кукса А. И., Платон Е. В. Об эффективности параллельных макроконвейерных вычислений в частично–блочных задачах линейного и булева линейного программирования // Кибернетика. 1986. № 4. С. 1–7.

47. Сергиенко И. В., Рощин В. А. Решение задачи разбиения множества с использованием параллельных вычислений // Системы программного обеспечения решения задач оптимального планирования. 1986. С. 104–105.

48. Сергиенко И. В., Гуляницкий Л. Ф. Фронтальные алгоритмы оптимизации для многопроцессорных ЭВМ // Кибернетика. 1981. № 6. С. 1–4.

49. Хохлюк В. И. Параллельные алгоритмы целочисленной оптимизации. М.: Радио и связь, 1987. 224 с.

50. Aida K., Natsume W., Futakata Y. Distributed computing with hierarchical master–worker paradigm for parallel branch and bound algorithm // Proceedings of Third Int. Symposium on Cluster Computing and Grid CCGRID. 2003. P. 156–163.

51. Solving large quadratic assignment problems on computational grids / K. M. Anstreicher, N. W. Brixius, J.-P. Goux et al. // Math. Programming. 2002. Vol. 91. P. 563–588.

52. Antonio J. K., Tsal W. K., Huang G. M. A highly parallel algorithm for multistage optimization problems and shortest path problems // Journal of Parallel and Distributed Computing. 1991. Vol. 12. P. 213–222.

53. Batoukov R., Soverik T. A generic parallel branch-and-bound environment on a network of workstations // Proceedings of Hiper-99. 1999. P. 474–483.

54. Parallel Mixed Integer Programming / R. Bixby, W. Cook, A. Cox et al. Rice University: Center for Research on Parallel Computation Research. Monograph CRPC-TR95554, 1995.

55. Recent trends in combinatorial optimization / R. Bixby, W. Cook, A. Cox et al. // Annals of Operations Research. 1999. Vol. 90. P. 19–43.

56. Casti J. Dynamic programming // Optimization Theory and Applications. 1973. no. 4. P. 423–438.

57. Chen Q., Ferris M. C., Linderoth J. T. Fatcop 2.0: Advanced features in an opportunistic mixed integer programming solver // Annals of Operations Research. 2001. Vol. 103. P. 17–32.

58. Eckstein J. Parallel branch-and-bound algorithms for general mixed integer programming on the CM-5 // SIAM J. Optimization. 1994. Vol. 4. P. 794–814.

59. Eckstein J., Phillips C. A., Hart W. E. PICO: An Object-Oriented Framework for Parallel Branch and Bound // RUTCOR Research Report 40-2000. Rutgers University, Piscataway, 2000.

60. Ferris M. C., Maravelias C. T., Sundaramoorthy A. Using grid computing to solve hard planning and scheduling problems // Proceedings of 18th European Symposium on Computer-Aided Process Engineering (ESCAPE 18), Lyon, France, June 1-4. 2008.

61. Gendron B., Crainic T. G. Parallel branch and bound algorithms: Survey and synthesis // Operations Research. 1994. Vol. 42. P. 1042–1066.

62. Goux J.-P., Leyffer S. Solving large MINLPs on computational grids // Optimization and Engineering. 2002. Vol. 3. P. 327–354.

63. Grama A., Kumar V. Parallel search algorithms for discrete optimization problems // ORSA Journal on Computing. 1995. Vol. 7. P. 365–385.

64. Grama A., Kumar V. State of the art in parallel search techniques for discrete optimization problems // IEEE Transactions on Knowledge and Data Engineering. 1999. Vol. 11. P. 28–35.

65. Ladanyi L., Ralphs T. K., Trotter L. E. Branch, cut, and price: Sequential and parallel // Computational Combinatorial Optimization. 2001. P. 223–260.

66. A grid–aware MIP solver: Implementation and case studies / E. P. Mancini, S. Marcarelli, I. Vasilyev et al. // Future Generation Computer Systems. 2008. Vol. 24. P. 133–141.

67. Ralphs T. K., Ladanyi L., Salzman M. J. Parallel branch, cut and price for large–scale discrete optimization // Mathematical Programming. 2003. P. 253–280.

68. Ralphs T. K. Parallel Branch and Cut / Ed. by E. Talbi. 2006. P. 53–101.

69. Roucairol C. Parallel processing for difficult combinatorial optimization problems // European Journal of Operational Research. 1996. Vol. 92. P. 573 – 590.

70. Shinano Y., Fujie T. ParaLEX: A parallel extension for the CPLEX mixed integer optimizer // Recent Advances in Parallel Virtual Machine and Message Passing Interface / Ed. by J. D. F. Cappello, T. Herault. Springer, 2007. P. 97–106.

71. Computational experience with a software framework for parallel integer programming / Y. Xu, T. K. Ralphs, L. Ladanyi et al. // The INFORMS Journal on Computing. 2009. Vol. 21. P. 383–397.

72. Щербина О. А. Локальные элиминационные алгоритмы решения разреженных дискретных задач // Ж. вычисл. матем. и матем. физ. 2008. Т. 1, № 48. С. 159–175.

73. Щербина О. А. Древовидная декомпозиция и задачи дискретной оптимизации (обзор) // Кибернетика и системный анализ. 2007. № 4. С. 102–118.

74. Bertele U., Brioschi F. Nonserial Dynamic Programming. New York: Academic Press, 1972. 235 p.

75. Щербина О. А. Локальные алгоритмы и древовидная декомпозиция для задач дискретной оптимизации // Динамические системы. 2006. № 20. С. 89–103.

76. Dechter R. Bucket elimination: A unifying framework for reasoning // Artificial Intelligence. 1999. Vol. 113. P. 41–85.

77. Лемтюжникова Д. В., Щербина О. А. Некоторые аспекты распараллеливания локального элиминационного алгоритма // Таврический вестник информатики и математики. 2012. Т. 1. С. 56–65.

78. Щербина О. А. Об одном локальном алгоритме для задач целочисленной оптимизации // Ж. вычисл. матем. и матем. физ. 1980. Т. 3, № 20. С. 802–804.

79. Щербина О. А. Асимптотические оценки эффективности локального алгоритма в дискретном программировании // Журн. вычисл. математики и матем. физики. 1982. Т. 22, № 6. С. 1360–1366.

80. Журавлев Ю. И., Финкельштейн Ю. Ю. Локальные алгоритмы для задач линейного целочисленного программирования // Проблемы кибернетики. 1965. № 14. С. 289–296.

81. Щербина О. А. Локальные элиминационные алгоритмы для разреженных задач дискретной оптимизации: дис. ... доктора физ.–мат. наук: 05.13.17. М., 2011. 361 с.

82. Финкельштейн Ю. Ю. Методы решения некоторых дискретных задач математического программирования: Автореф. дис. ... канд. физ.-мат. наук. М., 1966. 110 с.

83. Лемтюжникова Д. В., Щербина О. А. Локальный элиминационный алгоритм и параллельные вычисления // Материалы X Международной конференции [«Интеллектуальные системы и компьютерные науки»](Москва, 5–10 декабря). М.: МГУ им. М.В. Ломоносова, 2011. С. 357–360.

84. Лемтюжникова Д. В., Щербина О. А. Распараллеливание локального элиминационного алгоритма в блочных задачах дискретной оптимиза-

ции // Материалы V Международной конференции [«Танаевские чтения»] (Минск, 28–29 марта). Минск: ОИПИ НАН Беларуси, 2012. С. 55–60.

85. Лемтюжникова Д. В., Свириденко А. В. Вычислительные аспекты реализации локального элиминационного алгоритма для разреженных задач дискретной оптимизации // Материалы XIX Международной конференции [«Ломоносов 2012»] (Москва, 9–13 апреля) / под ред. А.И. Андреева, А.В. Андриянова, Е.А. Антипова [и др.]. М.: МАКС Пресс, 2012. URL: http://lomonosov–msu.ru/archive/Lomonosov_2012/1793/45987_4cce.pdf.

86. Лемтюжникова Д. В., Щербина О. А. Некоторые аспекты распараллеливания локального элиминационного алгоритма для задач дискретной оптимизации // Материалы V Всероссийской конференции [«Проблемы оптимизации и экономические приложения»] (Омск, 02–06 июля). 2012. с. 145.

87. Lemtyuzhnikova D., Shcherbina O. Parallel Local Elimination Algorithms for Sparse Discrete Optimization Problems // 12th International Conference ["Parallel Problem Solving From Nature (PPSN)"] (Taormina, 1–5 September). 2012. URL: http://neo.lcc.uma.es/workshops/PPSN2012/papers/p2.pdf.

88. Лемтюжникова Д. В., Свириденко А. В., Щербина О. А. Алгоритм выделения блочно–древовидной структуры в разреженных задачах дискретной оптимизации // Таврический вестник информатики и математики. 2012. Т. 1. С. 44–55.

89. Lemtyuzhnikova D., Shcherbina O. Parallel Local Elimination Algorithms for Sparse Discrete Optimization Problems. 2012. URL: http://neo.lcc.uma.es/workshops/PPSN2012/papers/p2.pdf.

90. Lemtyuzhnikova D., Sviridenko A., Shcherbina O. On local elimination algorithms for sparse discrete optimization problems // IV International Conference [Problems of Cybernetics and Informatics (PCI)], (Baku, 12–14 September). 2012. P. 1–4. URL: http://neo.lcc.uma.es/workshops/PPSN2012/papers/p2.pdf.

91. Lemtyuzhnikova D., Sviridenko A., Shcherbina O. On improvement of local algorithms for discrete optimization problems // IV International Conferece on Optimization Methods and Applications ["Optimization

and applications"(OPTIMA–2013)](Montenegro, September 22–28) / Ed. by V. Malkova. 2013. P. 106–107.

92. Лемтюжникова Д. В., В.Свириденко А., Щербина О. А. Стратегии повышения эффективности локального элиминационного алгоритма // Материалы Международной конференции [«Современная информатика: проблемы, достижения, и перспективы развития»], (Киев, 12–13 сентября). К: Институт кибернетики имени В.М.Глушкова НАН Украины, 2013. с. 8.

93. Лемтюжникова Д. В., Щербина О. А. Локальный элиминационный алгоритм и параллельные вычисления // Интеллектуальные системы. 2013. Т. 17, № 1–4. С. 490–494.

94. Лемтюжникова Д. В. Параллельное представление локального элиминационного алгоритма для решения разреженных задач дискретной оптимизации // Материалы 6–й международной конференции «Распределенные вычисления и Грид–технологии в науке и образовании», (Дубна, 30 июня — 5 июля). 2014.

95. Лемтюжникова Д. В. Параллельное представление локального элиминационного алгоритма для решения разреженных задач дискретной оптимизации // Компьютерные исследования и моделирование. 2015. Т. 7, № 3. С. 680–686.

96. Лемтюжникова Д. В. Модификация локального элиминационного алгоритма для эффективного решения больших разреженных задач дискретной оптимизации // Математические методы распознавания образов: 17–ая Всеросс. конф. М.: Торус, 2015. С. 108–109.

97. Д. В. Лемтюжникова Д.В. Ковков. Декомпозиция в многомерных задачах с разреженными матрицами // Известия РАН: Теория систем и управления. 2018. № 1.

98. Д. В. Лемтюжникова Д.В. Ковков. Тестирование алгоритмов для целочисленных квазиблочных задач оптимизации // Вестник БМТУ, серия Информационные технологии. 2017. № 6.

99. Д. В. Лемтюжникова Д.В. Ковков. Задачи дискретной оптимизации с квазиблочными матрицами // International Journal of Open Information Technologies. 2017. № 8.

100. Д. В. Лемтюжникова В.В. Волошинов В.И. Цурков. Распараллеливание на grid задач дискретной оптимизации с матрицами квазиблочной структуры // Известия РАН: Теория систем и управления. 2017. № 6.

101. Лемтюжникова Д. В. Декомпозиция разреженных матриц в задачах целочисленного программирования // Математические методы распознавания образов: 19–ая Всеросс. конф. М.: Торус, 2017. С. 56–57.

102. Duff I. S., Erisman A. M., Reid J. K. Direct methods for sparse matrices. Oxford University Press, 2017.

103. Fox L. An introduction to numerical linear algebra. London: Oxford University Press, 1964.

104. Wilkinson J. H. The algebraic eigenvalue problem. London: Oxford University Press, 1965.

105. Forsythe G., Moler C. B. Computer solutiOn of linear algebraic equations. New Jersey: Prentice–Hall, Englewood Cliffs, 1967.

106. Stewart G. Introduction to matrix computations. Academic Press, New York and London, 1973.

107. Strang G. Linear algebra and its applications (second edition). Academic Press, New York and London, 1980.

108. Golub G. H., Loan C. F. V. Matrix computations. Oxford: North Oxford Academic and Baltimore: John Hopkins Press, 1983.

109. Баландин М.Ю., Шурина Э.П. Методы решения СЛАУ большой размерности. Новосибирск: Изд-во НГТУ, 2000. 70 с.

110. Джордж А., Лю Дж. Численное решение больших разрежённых систем уравнений. М.: Мир, 1984. 333 с.

111. Воеводин В.В., Кузнецов Ю.А. Матрицы и вычисления. М.: Наука, 1984. 450 с.

112. Жигульская В. Ю. Численные методы. Луганск: Альма–матер, 2005. 137 с.

113. Tarjan R. E. Depth first search and linear graph algorithms // SIAM J. Comput. 1972. Vol. 1. P. 146–160.

114. Sargent R., Westerberg A. Speed-up in chemical engineering design // Trans. Inst. Chem. Eng. 1964. Vol. 42. P. 190–197.

115. Писсанецки С. Технология разрежённых матриц. М.: Мир, 1988. 410 с.

116. Murthy D., Anderson K. Sub–optimal control of sparsely coupled systems // Int. J. Syst. Sci. 1975. Vol. 6, no. 6. P. 565–578.

117. George J. A., Liu J. Computer solution of large sparse positive definite systems. Englewood Cliffs: Prentice-Hall Inc, 1981.

118. Parter S. The use of linear graphs in Gauss elimination // SIAM Review. 1961. Vol. 3. P. 119–130.

119. Rose D. J. A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations // Graph Theary and Computing. 1972. P. 183–217.

120. Liu J. Computational models and task scheduling for parallel sparse Cholesky factorization // Parallel Comput. 1986. Vol. 3. P. 327–342.

121. Two Dimensional Feature Selection by Sparse Matrix Regression / C. Hou, Y. Jiao, F. Nie et al. // IEEE Transactions on Image Processing. 2017.

122. Semi-External Memory Sparse Matrix Multiplication for Billion-Node Graphs / D. Zheng, D. Mhembere, V. Lyzinski et al. // IEEE Transactions on Parallel and Distributed Systems. 2017. Vol. 28, no. 5. P. 1470–1483.

123. Reducing the Cold–User and Cold-Item Problem in Recommender System by Reducing the Sparsity of the Sparse Matrix and Addressing the Diversity-Accuracy Problem / K. Bindu, R. L. Visweswaran, P. Sachin et al. // Proceedings of International Conference on Communication and Networks / Springer. 2017. P. 561–570.

124. Non-Parametric Sparse Matrix Decomposition for Cross-View Dimensionality Reduction / H. Liu, L. Liu, T. D. Le et al. // IEEE Transactions on Multimedia. 2017.

125. Regularized Non-negative Matrix Factorization for Identifying Differential Genes and Clustering Samples: a Survey / J.-X. Liu, D. Wang, Y.-L. Gao et al. // IEEE/ACM Transactions on Computational Biology and Bioinformatics. 2017.

126. Mesh Partitioning and Efficient Equation Solving Techniques by Distributed Finite Element Methods: A Survey / S. U. Ansari, M. Hussain, S. Mazhar et al. // Archives of Computational Methods in Engineering. 2017. P. 1–16.

127. A distributed approach for accelerating sparse matrix arithmetic operations for high-dimensional feature selection / A. Tommasel, D. Godoy, A. Zunino et al. // Knowledge and Information Systems. 2017. Vol. 51, no. 2. P. 459–497.

128. Elafrou A., Goumas G., Koziris N. Performance Analysis and Optimization of Sparse Matrix-Vector Multiplication on Intel Xeon Phi // Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2017 IEEE International / IEEE. 2017. P. 1389–1398.

129. Computing Urban Radiation: A Sparse Matrix Approach / J. P. Aguerre, E. Fernandez, G. Besuievsky et al. // Graphical Models. 2017.

130. Sympiler: Transforming Sparse Matrix Codes by Decoupling Symbolic Analysis / K. Cheshmi, S. Kamil, M. M. Strout et al. // arXiv preprint arXiv:1705.06575. 2017.

131. Sparse Matrix–Vector Multiplication on GPGPUs / S. Filippone, V. Cardellini, D. Barbieri et al. // ACM Transactions on Mathematical Software (TOMS). 2017. Vol. 43, no. 4. p. 30.

132. Hussain M., Kharat G. Person Detection and Tracking Using Sparse Matrix Measurement for Visual Surveillance // Proceedings of the International Conference on Data Engineering and Communication Technology / Springer. 2017. P. 281–293.

133. Salient object detection via structured matrix decomposition / H. Peng, B. Li, H. Ling et al. // IEEE transactions on pattern analysis and machine intelligence. 2017. Vol. 39, no. 4. P. 818–832.

134. Increasing the Efficiency of Sparse Matrix-Matrix Multiplication with a 2.5 D Algorithm and One-Sided MPI / A. Lazzaro, J. VandeVondele, J. Hutter et al. // arXiv preprint arXiv:1705.10218. 2017.

135. Rahmani M., Atia G. K. High dimensional low rank plus sparse matrix decomposition // IEEE Transactions on Signal Processing. 2017. Vol. 65, no. 8. P. 2004–2019.

136. Low Rank and Sparse Matrix Decomposition as Stroke Segmentation Prior: Useful or Not? A Random Forest-Based Evaluation Study / R. Werner, D. Schetelig, T. Sothmann et al. // Bildverarbeitung für die Medizin 2017. Springer, 2017. P. 161–166.

137. Sparse matrix factorizations for fast linear solvers with application to Laplacian systems / M. T. Schaub, M. Trefois, P. V. Dooren et al. // SIAM Journal on Matrix Analysis and Applications. 2017. Vol. 38, no. 2. P. 505–529.

138. Decomposition into low-rank plus additive matrices for background/foreground separation: A review for a comparative evaluation with a large-scale dataset / T. Bouwmans, A. Sobral, S. Javed et al. // Computer Science Review. 2017. Vol. 23. P. 1–71.

139. Data Compression for the Tomo-e Gozen Using Low-rank Matrix Approximation / M. Morii, S. Ikeda, S. Sako et al. // The Astrophysical Journal. 2017. Vol. 835, no. 1. p. 1.

140. Толковый словарь по теории графов в информатике и программировании / под ред. В. А. Евстигнеев, В. Н. Касьянов. Новосибирск: Наука, 1999. 288 с.

141. Зыков А. А. Основы теории графов. М.: «Вузовская книга», 2004. 664 с.

142. Щербина О. А. Локальные алгоритмы для блочно–древовидных задач дискретного программирования // Журн. вычисл. математики и матем. физики. 1985. Т. 25, № 8. С. 1143–1154.

143. Сигал И. Х., Иванова А. П. Введение в прикладное дискретное программирование. М: Физматлит, 2007. 304 с.

144. Авербах И. Л., Цурков В. И. Целочисленные оптимизационные модели блочного типа // Математическое моделирование. 1990. Т. 2, № 2. С. 39–57.

145. Тьюарсон Р. Разрежённые матрицы. М.: Мир, 1977. 191 с.

146. Rohat O., Popescu D. Is FBDK suitable for developing and implementing process control optimization problems? // System Theory, Control and Computing (ICSTCC), 18th International Conference. 2014. P. 117–122.

147. Chavan B., Udayakumar S. Tractor Transmission Validation for Synchronizer as Skid at Rig Level // Symposium on International Automotive Technology 17. 2017.

148. Ashcraft C., Liu J. Robust ordering of sparse matrices using multisection // SIAM J. Matrix Anal. Appl. 1995. Vol. 19. P. 816–832.

149. Pellegrini F., Roman J. Sparse matrix ordering with SCOTCH // Proceedings of HPCN'97, Vienna, LNCS 1225. 1997. P. 370–378.

150. Dantzig G. B. Programming of interdependent activities II: Mathematical model // Econometrica. 1949. Vol. 17. P. 200–211.

151. Ho J. K., Loute E. A set of staircase linear programming test problems // Mathematical Programming. 1981. no. 20. P. 245–250.

152. The Temporal Knapsack Problem and its solution / M. Bartlett, A. M. Frisch, Y. Hamadi et al. // II International Conference on the Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR). Vol. 2. 2005. P. 34–48.

153. Мезон Р., Фламгольц Э. Управление трудовыми ресурсами // Исследование операций. 1981. Т. 2: Модели и применения. С. 34–50.

154. Иванилов Ю. П., Пропой А. И. О задачах линейного динамического программирования // Докл. АН СССР. 1971. № 5. С. 1011–1014.

155. Оптимизация развития топливно-энергетического комплекса / под ред. А. С. Некрасов. М.: Энергоиздат, 1981. 240 с.

156. Laesanklang W., Landa-Silva D., Castillo-Salazar J. Mixed integer programming with decomposition to solve a workforce scheduling and routing problem // In: International Conference on Operations Research and Enterprise Systems (ICORES 2015), 10–12 January 2015. 2015. P. 283–293.

157. Heinz S., Beck J. C. Reconsidering Mixed Integer Programming and MIP-Based Hybrids for Scheduling // Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2012) / Ed. by N. Beldiceanu, N. Jussien, E. Pinson. 2012. P. 122–138.

158. Ciré A., Coban E., Hooker J. Mixed Integer Programming vs. Logic–Based Benders Decomposition for Planning and Scheduling // CPAIOR 2013: Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. 2013. P. 325–331.

159. Park J., Karumanchi S., Iagnemma K. Homotopy–Based Divide–and–Conquer Strategy for Optimal Trajectory Planning via Mixed-Integer Programming // IEEE Transactions on Robotics. 2015. Vol. 31, no. 5. P. 1101–1115.

160. Gade D., Küçükyavuz S., Sen S. Decomposition algorithms with parametric Gomory cuts for two–stage stochastic integer programs // Mathematical Programming. 2014. Vol. 144, no. 1. P. 39–64.

161. Ntaimo L. Fenchel decomposition for stochastic mixed–integer programming // Journal of Global Optimization. January, 2013. Vol. 55, no. 1. P. 141–163.

162. Chu Y., You F. Integration of production scheduling and dynamic optimization for multi–product CSTRs: Generalized Benders decomposition coupled with global mixed–integer fractional programming // Computers & Chemical Engineering. 2013. Vol. 58. P. 315–333.

163. An Effective Problem Decomposition Method for Scheduling of Diffusion Processes Based on Mixed Integer Linear Programming / C. Jung, D. Pabst, M. Ham et al. // IEEE Transactions on Semiconductor Manufacturing. 2014. Vol. 27, no. 3. P. 357–363.

164. Luedtke J. A branch–and–cut decomposition algorithm for solving chance–constrained mathematical programs with finite support // Mathematical Programming. 2014. Vol. 146, no. 1. P. 219–244.

165. Integration of progressive hedging and dual decomposition in stochastic integer programs / G. Guo, G. Hackebeil, S. M. Ryan et al. // Operations Research Letters. 2015. Vol. 43, no. 3. P. 311–316.

166. Mitra S., Garcia-Herreros P., Grossmann I. E. A Novel Cross–decomposition Multi–cut Scheme for Two–Stage Stochastic Programming // Computer Aided Chemical Engineering. 2014. Vol. 32. P. 241–246.

167. Popovic Z., Kerleta V. D., Popovic D. Hybrid simulated annealing and mixed integer linear programming algorithm for optimal planning of radial distribution networks with distributed generation // Electric Power Systems Research. 2014. Vol. 108. P. 211–222.

168. A MILP (Mixed Integer Linear Programming) decomposition solution to the scheduling of heavy oil derivatives in a real–world pipeline / J. A. Fabro, S. L. Stebel, D. Rossato et al. // Computers & Chemical Engineering. 2014. Vol. 66, no. 3. P. 124–138.

169. A Base Integer Programming Model and Benchmark Suite for Liner-Shipping Network Design / B. D. Brouer, J. F. Alvarez, C. E. M. Plum et al. // Transportation Science. 2014. Vol. 48, no. 2.

170. Network–Constrained AC Unit Commitment Under Uncertainty: A Benders' Decomposition Approach / A. Nasri, S. J. Kazempour, A. J. Conejo et al. // IEEE Transactions on Power Systems. 2016. Vol. 31, no. 1. P. 412–422.

171. Hertz A., Montagne R., Gagno F. Integer programming models for the partial directed weighted improper coloring problem // Journal of Graph Algorithms and Applications. 2016. Vol. 20, no. 2. P. 159–188.

172. Ciré A., Coban E., Hooker J. Logic-based Benders decomposition for planning and scheduling: a computational analysis // Constraint Satisfaction for Planning and Scheduling. 2016. Vol. 31, no. 5. P. 440–451.

173. Bodur M., Luedtke J. R. Mixed–Integer Rounding Enhanced Benders Decomposition for Multiclass Service-System Staffing and Scheduling with Arrival Rate Uncertainty // Management Science. 2014. URL: http://www.optimization–online.org/DB_FILE/2013/10/4080.pdf.

174. Fazlollahi S., Marechal F. Multi–objective, multi–period optimization of biomass conversion technologies using evolutionary algorithms and mixed integer linear programming (MILP) // Applied Thermal Engineering. 2013. Vol. 50, no. 2. P. 1501–1513.

175. Demirel E., Demirel N., Gokcen H. A mixed integer linear programming model to optimize reverse logistics activities of end-of-life vehicles in Turkey // Journal of Cleaner Production. 2016. Vol. 112. P. 2101–2113.

176. Xu P., Wang L. An exact algorithm for the bilevel mixed integer linear programming problem under three simplifying assumptions // Computers & Operations Research. 2014. Vol. 41. P. 309–318.

177. Ambrosino D., Paolucci M., Sciomachen A. Experimental evaluation of mixed integer programming models for the multi-port master bay plan problem // Flexible Services and Manufacturing Journal. 2015. Vol. 27, no. 2. P. 263–284.

178. A Multi-stage Decomposition Heuristic for the Container Stowage Problem / M. Gumus, P. Kaminsky, E. Tiemroth et al. // M&SOM 2008 Conference Proceedings, June 5-6, 2008. 2008.

179. Tang X., Blandin S., Wynter L. A fast decomposition approach for traffic control // IFAC Proceedings Volumes. 2014. Vol. 47, no. 3. P. 5109–5114.

180. Obtaining lower bounds from the progressive hedging algorithm for stochastic mixed–integer programs / D. Gade, G. Hackebeil, S. M. Ryan et al. // Mathematical Programming. 2016. Vol. 157, no. 1. P. 47–67.

181. On parallelizing dual decomposition in stochastic integer programming / M. Lubin, K. Martin, C. G. Petra et al. // Operations Research Letters. 2013. Vol. 41, no. 3. P. 252–258.

182. Zhang M., Küçükyavuz S. Finitely Convergent Decomposition Algorithms for Two-Stage Stochastic Pure Integer Programs // SIAM J. Optim. 2014. Vol. 24, no. 4. P. 1933–1951.

183. Liu X., Küçükyavuz S., Luedtke J. Decomposition algorithms for two-stage chance-constrained programs // Mathematical Programming. 2016. Vol. 157, no. 3. P. 219–243.

184. Using Benders Decomposition for Solving Ready Mixed Concrete Dispatching Problems / M. Maghrebi, V. Periaraj, S. T. Waller et al. // The 31st International Symposium on Automation and Robotics in Construction and Mining (ISARC 2014). 2014.

185. Mitra S., Garcia-Herreros P., Grossmann I. E. A cross–decomposition scheme with integrated primal–dual multi–cuts for two–stage stochastic programming investment planning problems // Mathematical Programming. 2016. Vol. 157, no. 1. P. 95–119.

186. A Lagrangian decomposition approach for the pump scheduling problem in water networks / B. Ghaddar, J. Naoum-Sawaya, A. Kishimoto et al. // European Journal of Operational Research. 2015. Vol. 241, no. 2. P. 490–501.

187. Журавлев Ю. И. Избранные научные труды. М.: Магистр, 1998. 420 с.

188. Robertson N., Seymour P. Graph minors II. Algorithmic aspects of treewidth // J. Algorithms. 1986. Vol. 7. P. 309–322.

189. Tarjan R. E. Simple linear–time algorithms to test chordality of graphs, test acyclity of hypergraphs, and selectively reduce acyclic hypergraphs // SIAM J. Comput. 1984. Vol. 13. P. 566–579.

190. Arnborg S., Corneil D. G., Proskurowski A. Complexity of finding embeddings in a k-tree // SIAM J. Alg. Disc. Meth. 1987. Vol. 8. P. 277–284.

191. Yannakakis M. Computing the minimum fill-in is NP–complete // SIAM J. Alg. Disc. Meth. 1981. Vol. 2. P. 77–79.

192. Liu J. Modification of the minimum-degree algorithm by multiple elimination // ACM Transactions on Mathematical Software. 1985. Vol. 11. P. 141–153.

193. Amestoy P. R., Davis T. A., Duff I. S. An approximate minimum degree ordering algorithm // SIAM Journal on Matrix Analysis and Applications. 1996. Vol. 17, no. 4. P. 886–905.

194. George J. A. Nested dissection of a regular finite element mesh // SIAM J. Numer. Anal. 1973. Vol. 10. P. 345–367.

195. Hendrickson B., Rothberg E. Improving the run time and quality of nested dissection ordering // SIAM J. Sci. Comput. 1998. Vol. 20(2). P. 468–489.

196. Jégou P., Ndiaye S. N., Terrioux C. Computing and exploiting tree–decompositions for (Max–)CSP // Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming (CP-2005). 2005. P. 777–781.

197. Rose D. J., Tarjan R., Lueker G. Algorithmic aspects of vertex elimination on graphs // SIAM J. on Computing. 1976. Vol. 5. P. 266–283.

198. Щербина О. А. Постоптимизационный анализ в дискретном программировании (обзор) // Методы оптимизации в экономико-математ. 1988. С. 28–50.

199. Emelichev V., Korotkov V. Post–optimal analysis of investment problem with Wald's ordered maximin criteria // Matematica. 2012. Т. 1, № 68. С. 59–69.

200. Pisinger D., Saidi A. Tolerance analysis for 0–1 knapsack problems // European Journal of Operational Research. 2016. Vol. 40, no. 1. P. 475–489.

201. Goberna M. A., Lopez M. A. Post–Optimal Analysis in Linear Semi-Infinite Optimization. Springer, 2014.

202. Geffken S., Buskens C. Feasibility refinement in sequential quadratic programming using parametric sensitivity analysis // Optimization Methods and Software. 2016. P. 1–16.

203. Финкельштейн Ю. Ю. Приближенные методы и прикладные задачи дискретного программирования. М.: Наука, 1976. 265 с.

204. Geoffrion A. M. Lagrangean relaxation and its uses in integer // Math. Stud. 1974. P. 82–114.

205. Gorry G. A., Shapiro J. F., Wolsey L. A. Relaxation methods for pure and mixed integer programming problems // Management Science. 1972. Vol. 18, no. 5. P. 229–239.

206. Discrete Optimization with Decision Diagrams / D. Bergman, A. A. Cire, W.-J. van Hoeve et al. // INFORMS Journal on Computing. 2016. Vol. 28, no. 1. P. 47–66.

207. On linear conic relaxation of discrete quadratic programs / T. Nie, S.-C. Fang, Z. Deng et al. // Optimization Methods and Software. 2016. Vol. 31, no. 4. P. 737–754.

208. Forrest J., Tomlin J. A. Branch and Bound, Integer, and Non-integer Programming // Annals of Operations Research. 2007. P. 81–87.

209. Bader D. A., Hart W. E., Phillips C. A. Parallel algorithm design for branch and bound / Ed. by H. Greenberg. Tutorials on Emerging Methodologies and Applications in Operations Research, Kluwer Academic Press, 2004. P. 1–44.

210. Aida K., Osumi T. A Case Study in Running a Parallel Branch and Bound Application on the Grid // In: SAINT 205: Proceedings of the the 2005 Symposium onApplications and the Internet. 2005. P. 164–173.

211. Efficient parallel branch-and-bound algorithm for constructing minimum ultrametric trees / K.-M. Yu, J. Zhou, C.-Y. Lin et al. // Journal of Parallel and Distributed Computing. 2009. Vol. 69, no. 11. P. 905–914.

212. Mezmaz M., Melab N., Talbi E. G. An efficient load balancing strategy for grid–based branch and bound algorithm // Parallel Computing. 2007. Vol. 33, no. 4–5. P. 302–313.

213. Ralphs T. K., Ladanyi L., Salzman M. J. A library hierarchy for implementing scalable parallel search algorithms // Journal of Supercomputing. 2004. Vol. 28, no. 2. P. 215–234.

214. Crainic T. G., Cun B. L., Roucairol C. Chapter 1. Parallel Branch–and–Bound Algorithms / Ed. by E.-G. Talbi. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2006.

215. Building a Parallel Branch and Bound Library. In Solving Combinatorial Optimization Problems in Parallel / M. Benchouche, V.-D. Cung, S. Dowaji et al. // Lecture Notes in Computer Science. 1996. Vol. 1054. P. 201–231.

216. Junger M., Thienel S. The ABACUS System for Branch and Cut and Price Algorithms in Integer Programming and Combinatorial Optimization // Software Practice and Experience. 2000. Vol. 30. 1325 p.

217. Junger M., Thienel S. Introduction to ABACUS–a branch–and–cut system // Operations Research Letters. 1998. Vol. 22. 83 p.

218. Kaan L., Olinick E. The Vanpool Assignment Problem: Optimization models and solution algorithms // Computers & Industrial Engineering. 2013. Vol. 66, no. 1. P. 24–40.

219. A mixed–integer linear programming approach for optimal type, size and allocation of distributed generation in radial distribution systems / A. C. Rueda-Medina, J. F. Franco, M. J. Rider et al. // Electric Power Systems Research. April, 2013. Vol. 97. P. 133–143.

220. Malawski M., Figiela K., Nabrzyski J. Cost minimization for computational applications on hybrid cloud infrastructures // Future Generation Computer Systems. September, 2013. Vol. 29, no. 7. P. 1786–1794.

221. Viana A., Pedroso J. A new MILP–based approach for unit commitment in power production planning // International Journal of Electrical Power & Energy Systems. January, 2013. Vol. 44, no. 1. P. 997–1005.

222. A mixed–integer LP model for the optimal allocation of voltage regulators and capacitors in radial distribution systems / J. F. Franco, M. J. Rider, M. Lavorato et al. // International Journal of Electrical Power & Energy Systems. June, 2013. Vol. 48. P. 123–130.

223. Noyan N., Rudolf G. Optimization with Multivariate Conditional Value–at–Risk Constraints // Operations Research. July-August, 2013. Vol. 61, no. 4. P. 990–1013.

224. Gentilini I., Margot F., Shimada K. The travelling salesman problem with neighbourhoods: MINLP solution // Optimization Methods & Software. 2013. Vol. 28, no. 2. P. 364–378.

225. Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers / J. Tordsson, R. Montero, R. Moreno-Vozmediano et al. // Future Generation Computer Systems. 2012. Vol. 28, no. 2. P. 358–367.

226. Foster I., Kesselman C. The Grid: Blueprint for a New Computing Infrastructure. San Francisco: Morgan Kaufmann Publishers Inc, CA, 1988.

227. Колпаков Р.М., Посыпкин М.А. Асимптотические характеристики метода ветвей и границ для задачи о ранце в распределенной вычислительной среде // Труды III Международной конференции "Системный анализ и информационные технологии"(САИТ - 2009) (сентябрь 2009 г., Звенигород, Россия). 2009. С. 708–716.

228. Посыпкин М. А., Сигал И. Х. Оценки ускорения для некоторых вариантов параллельной реализации метода ветвей и границ // Ж. вычисл. матем. и матем. физ. 2006. Т. 46, № 12. С. 2289–2304.

229. Посыпкин М. А., Хританков А. С. О понятии ускорения и эффективности в распределенных системах // Труды Всероссийской научной конференции «Научный сервис в сети Интернет: решение больших задач». 2008. С. 149–155.

230. Glankwamdee W., Linderoth J. T. MW: A Software Framework for Combinatorial Optimization on Computational Grids // Parallel Combinatorial Optimization. 2006. P. 239–261.

231. Aida K., Futakata Y., Osumi T. Parallel Branch and Bound Algorithm with the Hierarchical Master-Worker Paradigm on the Grid // IPSJ Digital Courier. 2006. Vol. 2. P. 584–591.

232. Mezmaz M., Melab N., Talbi E. G. A Grid–enabled Branch and Bound Algorithm for Solving Challenging Combinatorial Optimization Problems // 21th IEEE Intl. Parallel and Distributed Processing Symp., (Long Beach, California, Mar. 26-30). 2007.

233. Concurrent Data Structures and Load Balancing Strategies for Parallel Branch and Bound/A* Algorithms / V.-D. Cung, S. Dowaji, B. L. Cun et al. // DIMACS Series in Discrete Optimization and Theoretical Computer Science. 1997. Vol. 30. 141 p.

234. McCreesh C., Prosser P. The Shape of the Search Tree for the Maximum Clique Problem and the Implications for Parallel Branch and Bound // ACM Transactions on Parallel Computing. 2015. Vol. 2(1), no. 8.

235. Memory aware load balance strategy on a parallel branch-and-bound application / J. Silva, C. Boeres, L. Drummond et al. // Concurrency and Computation: Practice and Experience. 2015. Vol. 27, no. 5. P. 1122–1144.

236. A grid–enabled distributed branch–and–bound algorithm with application on the Steiner Problem in graphs / L. M. Drummond, E. Uchoa, A. D. Goncalves et al. // Operations Research. 2006. Vol. 32, no. 9. P. 629–642.

237. Owens J. D., Luebke D., Govindaraju N. A survey of general–purpose computation on graphics hardware // State of the Art Reports. 2005. P. 21–51.

238. Pedemonte M., Nesmachnow S., Cancela H. A survey on parallel ant colony optimization // Applied Soft Computing. 2011. P. 5181–5197.

239. Luong T.-V., Melab N., Talbi E.-G. Parallel Hybrid Evolutionary Algorithms on GPU // IEEE Congress on Evolutionary Computation (CEC). Barcelone, 2010.

240. Luong T.-V., Melab N., Talbi E.-G. GPU–based Multi–start Local Search Algorithms // Proceedings of Learning and Intelligent Optimization (LION'2011. 2011. P. 321–335.

241. Boyer V., El D. B., Elkihel M. Solving knapsack problems on GPU // Comput. Oper. Res. 2012. P. 42–47.

242. Fujimoto N., Tsutsui S. A highly–parallel TSP solver for a GPU computing platform // Proc. of the 7th int. conf. on Numerical methods and applications (NMA'10). 2010. P. 264–271.

243. Gulati K., Khatri S. P. Boolean Satisfiability on a Graphics Processor // ACM Great Lakes Symposium on VLSI. 2010. P. 123–126.

244. Allouche D., Givry S. D., Schiex T. Towards parallel non serial dynamic programming for solving hard weighted CS // Proceedings of the 16th international conference on Principles and practice of constraint programming (CP'10) / Ed. by D. Cohen. 2010. P. 53–60.

245. Otten L., Dechter R. Towards parallel search for optimization in graphical models // Proc. of ISAIM. 2010.

246. Otten L., Dechter R. Advances in distributed branch and bound // Proceedings of ECAI'12. 2012.

247. Колпаков Р.М., Посыпкин М.А., Сигал И.Х. О нижней оценке вычислительной сложности одной параллельной реализации метода ветвей и границ // Автоматика и телемеханика. 2010. № 10. С. 156–166.

248. Хохлюк В. И. Параллельные алгоритмы целочисленной оптимизации. 2007. URL: http://math.nsc.ru/LBRT/u1/hohl/book.pdf.

249. Mitra G., Hai I., Hajian M. T. A distributed processing algorithm for solving integer programs using a cluster of workstations // Parallel Computing. 1997. Vol. 23(6). P. 733–753.

250. Sergienko I. V., Shilo V. P., Roshchin V. A. Systems Analysis. Optimization parallelizing for discrete programming problems // Cybernetics and Systems Analysis. 2004. Vol. 40, no. 2. P. 184–189.

251. Bourbeau B., Crainic T. G., Gendron B. Branch–and–bound parallelization strategies applied to a depot location and container fleet management problem // Parallel Computing. 2000. Vol. 26, no. 1. P. 27–46.

252. Cun B. L., Roucairol C., Crainic T. G. Parallel Combinatorial Optimization. USA: John Wiley and Sons, 2006.

253. Колпаков Р.М., Посыпкин М.А. Об оценках вычислительной сложности параллельного варианта МВГ для серии задач о ранце // Известия Российской академии наук. Теория и системы управления. 2011. № 5. С. 74–82.

254. Koutsopoulos I., Papaioannou T. G., Hatzi V. Modeling and Optimization of the Smart Grid Ecosystem // Foundations and Trends in Networking. 2016. Vol. 10, no. 2–3. P. 115–316. URL: http://dx.doi.org/10.1561/1300000042.

255. Salah A., Hart E. A Modified Grid Diversity Operator for Discrete Optimization and its Application to Wind Farm Layout Optimization Problems // Companion Material Proceedings Genetic and Evolutionary Computation Conference, GECCO 2016 (Denver, CO, USA, July 20–24) / Ed. by T. Friedrich, F. Neumann, A. M. Sutton. 2016. P. 977–982.

256. Волошинов В. В., Неверов В. С., Смирнов С. А. Интеграция программных средств оптимизационного моделирования в неоднородной вычислительной среде на основе системы AMPLX // Электронный сборник тезисов докладов НСКФ'2015, Институт программных систем имени А.К. Айламазяна РАН (Москва, 24–27 ноября). 2015. с. 33. URL: http://2015.nscf.ru/TesisAll/4_Reshenie_zadach_optimizatsii/10_494_Voloshinc

Приложение А

Профили работы ЛБЭАП для задач с квазиблочной структурой

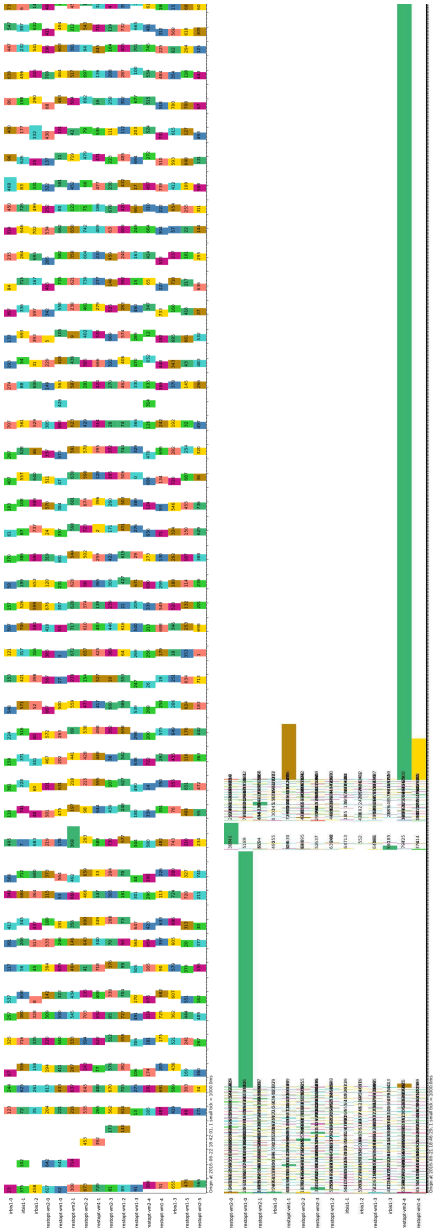


Рис. А. Профили работы ЛБЭАП для задачи с БД и БЛ структурами.