Модификации локального элиминационного алгоритма для эффективного решения больших разреженных задач дискретной оптимизации

Д.В.Лемтюжникова

Вычислительный центр им. А.А.Дородницына РАН

Светлогорск • 19–25 сентября 2015

Разреженная задача ЦЛП

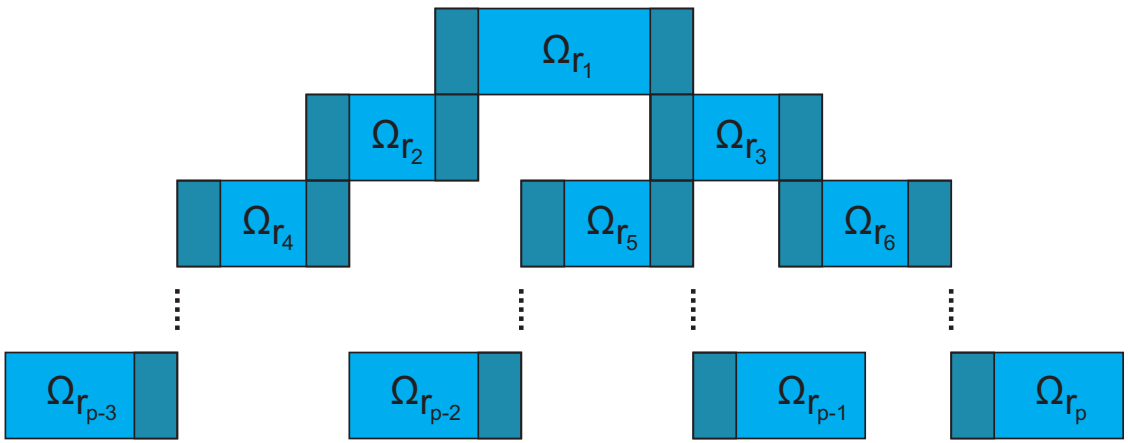
$$\begin{cases} c_1 x_1 + c_2 x_2 + \ldots + c_n x_n \to \min, \\ a_{11} x_1 + \ldots + a_{1n} x_n \geq b_1, \\ \ldots \\ a_{m1} x_1 + \ldots + a_{mn} x_n \geq b_m, \\ \{x_1; \ldots; x_n\} \in \{0; 1\}. \\ \Omega \subset (1, \ldots, n) \times (1, \ldots, m) \\ \Omega = \{(i, j) \mid a_{ij} \neq 0\} \end{cases}$$

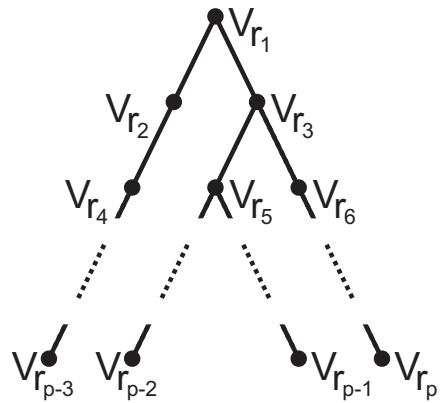
$z = |\Omega|$ — число ненулевых элементов

Разреженные задачи с блочно–древовидной структурой

Блочно–древовидная структура — блочная структура, в которой все внутренние блоки имеют связь как минимум с двумя другими блоками, а крайние — только с одним.
Ω — система окрестностей для каждого блока r_i.
Переменные x для задачи ДО:

- блоковая переменная $x_{r_i} \in \Omega_{r_i}$
- сепаратор $x_{r_{i_1},\ldots,r_{i_s}} \in \cup_{j=1}^{s} \Omega_{r_{i_j}}$





Граф пересечений — дерево

Локальный элиминационный алгоритм (ЛЭА), предложенный О.А. Щербиной:

Этап 1. Подготовка данных

Этап 2. Поиск локальных решений

Этап 3. Систематизация найденных решений

Щербина О.А. Локальные элиминационные алгоритмы для разреженных задач дискретной оптимизации // Докторская диссертация. — Москва, 2010. — 361 с.

Этап 1. Подготовка данных

Шаг 1. Препроцессинг

Шаг 2. Выделение БД структуры с помощью модифицированного алгоритма Финкельштейна

Шаг 3. Поиск оптимального порядка элиминации — исключения переменных соответствующих подзадач

Шаг 4. Анализ полученных решений с помощью постоптимального анализа

Этап 1. Шаг 1. Препроцессинг

- фаза между формулированием модели и ее решением, которая применяет простые логические правила (тесты) для упрощения задачи;
- может уменьшить размер задачи в результате фиксирования некоторых переменных и исключения ряда ограничений, а также выявить недопустимость задачи.

Этап 1. Шаг 2. Выделение блочно–древовидной структуры

Теорема

Чтобы $n \times m$-матрица A была блочно-древовидной с k блоками, необходимо, чтобы:

1) $n + 2m - z - 1 \leq k \leq m$;

2) $\frac{n+2m+p+3}{4} + \frac{\sqrt{(n-2m+p+3)^2+8(2n-m-z)}}{4} \leq k \leq \frac{n+1}{2}$;

где z — число ненулевых элементов матрицы,

p — степень дерева; $k \geq 2$; $m \geq 2$; $n \geq 3$; $4 \leq z < mn$;

$2 \leq p \leq k-1$, $k, m, n, z, p \in \mathbb{N}$.

Следствие

Степень дерева, соответствующего $n \times m$-матрице A:

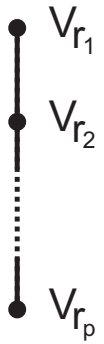
$p \geq \frac{2n-k-z}{k-m} - n + 2k - 2$.

Этап 1. Шаг 2. Выделение блочно–древовидной структуры

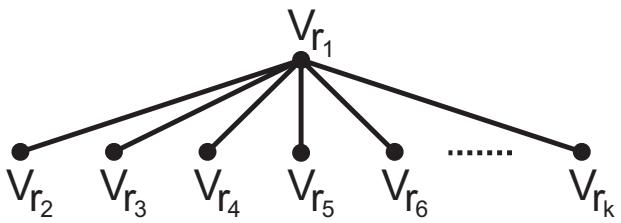
Частные случаи БД структуры

$z \le (m-k)(n-2k+2+p)+2n-k$

$2 \le p \le k-1$



квазиблочная структура



двухслойная структура

Максимальное число ненулевых элементов в матрице для нижней и верхней границы самой старшей степени дерева:

$$z_{\max} = \begin{cases} (m-k)(n-2k+4)+2n-k, & p=2 \\ (m-k)(n-k+1)+2n-k, & p=k-1 \end{cases}$$

Этап 1. Шаг 2. Выделение блочно-древовидной структуры

Модифицированная декомпозиция Финкельштейна:

- Выделение квазиблочной структуры в разреженной матрице [Ю.Ю.Финкельштейн]
- Выделение компонент связности в квазиблочной структуре [О.А.Щербина]
- Переход к блочно-древовидной структуре

Финкельштейн Ю. Ю. Методы решения некоторых дискретных задач математического программирования: Дисс. к.ф.-м.н., 1966. 110 с.

Щербина О. А. Локальные алгоритмы для блочно-древовидных задач дискретного программирования // ЖВМиМФ, 1985, Т.25, №8, 1143–1154.

Этап 1. Шаг 3. Нахождение оптимального порядка элиминации

Нахождение оптимального порядка элиминации
— NP–трудная задача
$\rightarrow$ определение порядка элиминации с помощью эвристик:

- 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)

Этап 1. Шаг 4. Постоптимальный анализ

- позволяет использовать при решении задач информацию, полученную при решении уже решенных задач того же пакета
- процедура ПА позволяет эффективно пересчитывать задачу ЦЛП при изменениях параметров задачи

Шаг 1. Выделение подзадач согласно порядку элиминации

Шаг 2. Решение подзадачи при помощи решателя

Шаг 3. Сохранение решения в таблицу

Шаг 1. **Выделение подзадач согласно порядку элиминации** — полный перебор по множествам переменных при достаточно больших сепараторах $\rightarrow$ модификации:

- ЛЭА с релаксациями
- ЛЭА с оракулом
- Параллельный ЛЭА

Шаг 2. Решение подзадачи при помощи решателя

Шаг 3. Сохранение решения в таблицу

Этап 2. Шаг 1. Выделение подзадач согласно порядку элиминации

Модификация 1. ЛЭА с релаксациями

Релаксации — построение оценочных задач:

- линейная релаксация:
  условия $x_j = \{0; 1\}$ заменяются неравенствами $0 \leq x_j \leq 1$
  (задача линейного программирования);
- ранцевая релаксация:
  линейная комбинация исходных ограничений вместо исходной системы ограничений.

Этап 2. Шаг 1. Выделение подзадач согласно порядку элиминации

Модификация 2. ЛЭА с оракулом

Оракул — процедура, позволяющая «предсказать» искомые оптимальные значения сепараторов.

Все наборы значений сепараторов перебираются полностью, но трудоемкость выполнения шагов снижена за счет решения релаксированных подзадач.

- Замена соответствующих блоков задач ДО на их релаксации
- Решение полученной задачи с помощью ЛЭА
- Нахождение значений сепараторов, близких к оптимальным
- Решение задач, соответствующих блокам, независимо друг от друга.

Этап 2. Шаг 1. Выделение подзадач согласно порядку элиминации

Модификация 3. Параллельный ЛЭА

Распараллеливание — процесс адаптации локального элиминационного алгоритма на вычислительной системе параллельной архитектуры для повышения его эффективности.

Источники распараллеливания:

- Древовидное распараллеливание — независимые ветви элиминационного дерева
- Блочное распараллеливание — распараллеливание по значениям сепаратора

Цель:

подготовка данных и улучшение вычислимости локального элиминационного алгоритма за счёт его различных модификаций

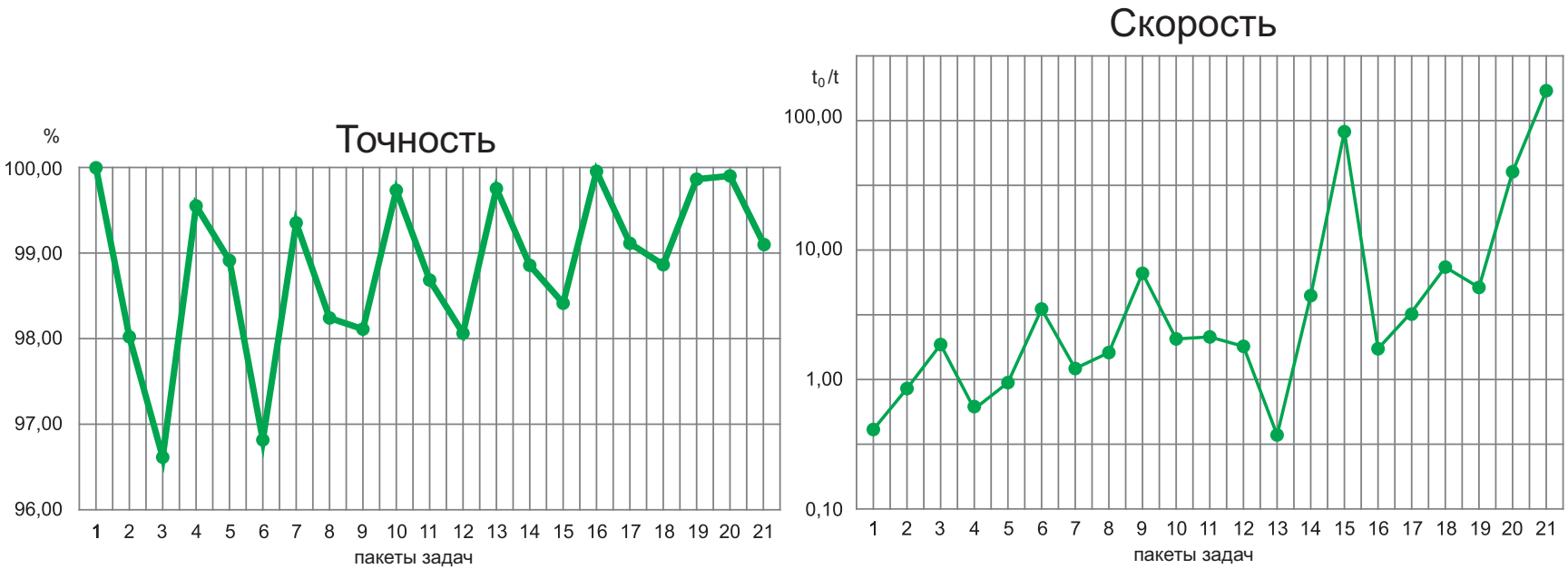
Технические характеристики:

- Библиотека LES (Local Elimination Solver) для задач бинарного целочисленного линейного программирования
- Процессор Intel Core i3 M 390 @ 2.67 ГГц
- Решатель — SYMPHONY версии 5.4.1
- Решатель — SCIP версии 3.0.2
- Решатель для релаксаций — CLP версии 3.0.2
- Грид — ICC KNU Cluster (КНУ, Киев)

Цель:
Показать влияние препроцессинга на скорость алгоритма

Описание эксперимента:

- Генерировались задачи с квазиблочной структурой на основе гиперграфов из библиотеки CSP проекта DBAI Hypertree Project
- Приближённая модификация ЛЭА



Результаты:

- Неэффективен для решения задач с блоками малой размерности и с небольшими сепараторами;
- Ускоряет алгоритм для задач большой размерности с большими сепараторами.

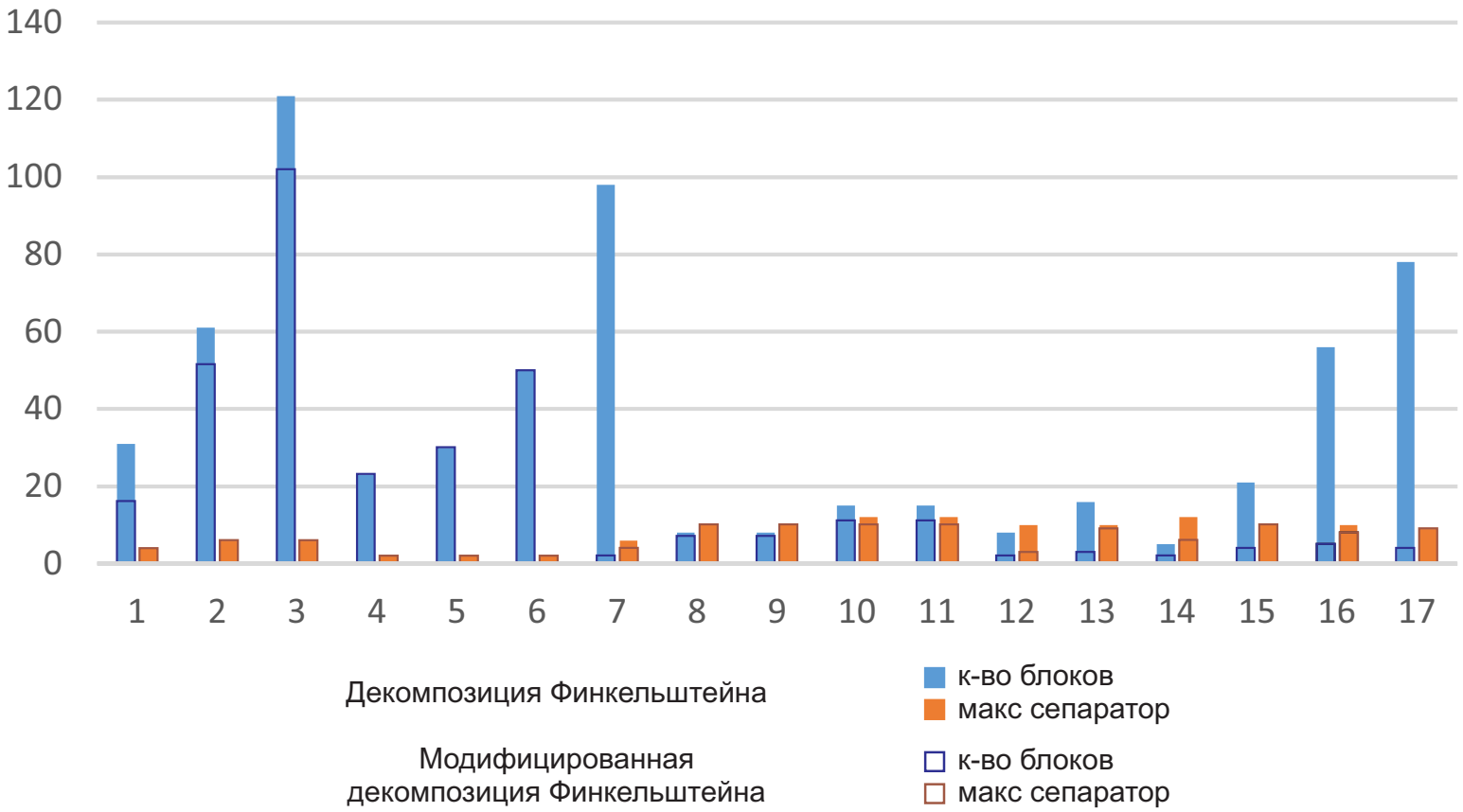
Этап 1. Шаг 2. Модифицированная декомпозиция Финкельштейна

Цель эксперимента:
выделение блочно–древовидной структуры

Описание эксперимента:

- Тестовые задачи генерировались на основе гиперграфов из библиотеки CSP проекта DBAI Hypertree Project.
- Каждая тестовая задача декомпозировалась с помощью исходного и модифицированного алгоритмов Финкельштейна
- При превышении допустимого размера сепаратора в блоке, выполнялась процедура слияния блоков

Этап 1. Шаг 2. Модифицированная декомпозиция Финкельштейна



Этап 1. Шаг 2. Модифицированная декомпозиция Финкельштейна

Результаты:

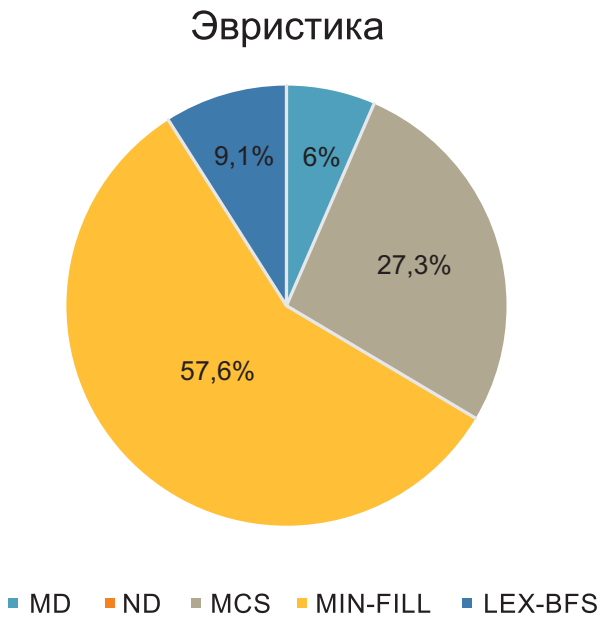
- При большом количестве блоков модифицированный алгоритм существенно уменьшает количество построенных блоков и размеры сепараторов
- При небольшом количестве блоков различия модифицированного алгоритма и алгоритма без модификации не существенны

Цель: создание списка переменных для их исключения из задачи оптимизации и составления соответствующих подзадач

Описание эксперимента:

- Тестовые задачи генерировались на основе гиперграфов из библиотеки CSP проекта DBAI Hypertree Project.
- Для каждого графа взаимосвязей применялись эвристики
- Каждая задача решалась с полученным порядком элиминации и сравнивалось время решения

Этап 1. Шаг 3. Порядок элиминации



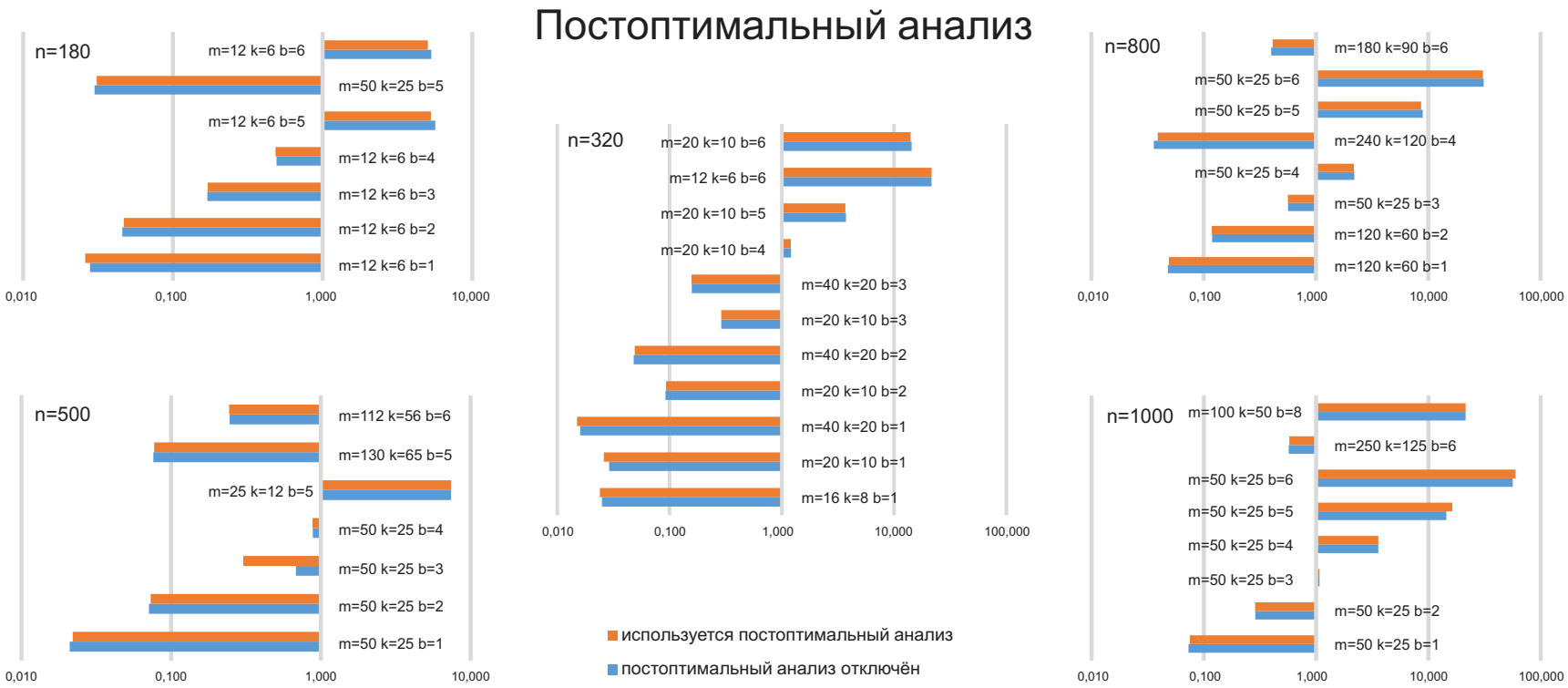
- Более чем в половине случаев лучшее время работы алгоритма получено в совокупности с алгоритмом минимального пополнения;
- Почти в трети случаях минимальное время достигается с помощью алгоритма лексикографического поиска в ширину.

Цель:
анализ полученных результатов для решения подобных задач

Описание эксперимента:

- Генерировались задачи на основе гиперграфов из библиотеки CSP проекта DBAI Hypertree Project.
- Величина сепаратора не превышала 8 переменных
- Число переменных не превышало 1000
- Сравнивались ЛЭА и ЛЭА+SYMPHONY (с встроенным постоптимальным анализом)

Этап 1. Шаг 4. Постоптимальный анализ



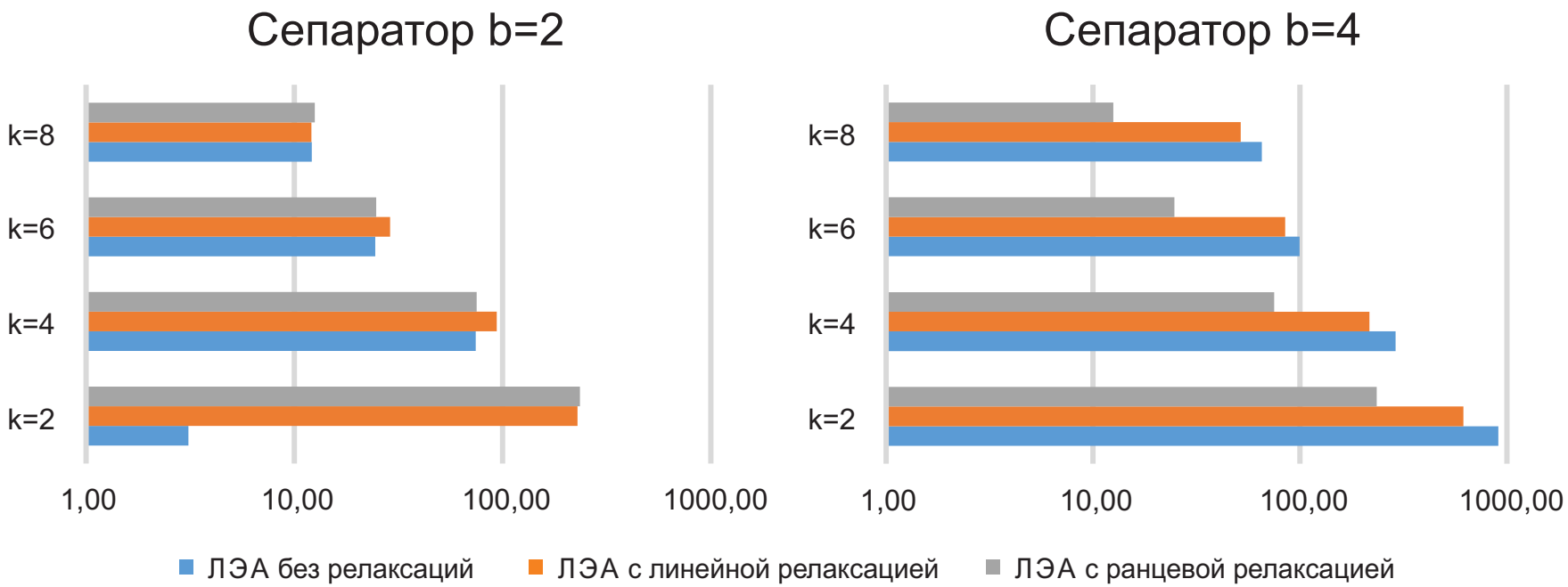
Алгоритм с постоптимальным анализом в большинстве случаев намного эффективнее решает однотипные задачи.

Цель: исследовать, насколько эффективно использовать релаксации для ЛЭА

Описание эксперимента:

- Генерировались задачи с квазиблочной структурой на основе гиперграфов из библиотеки CSP проекта DBAI Hypertree Project.
- Препроцессинг отключен
- Применено три теста на допустимость решения

m=100, n=100



При больших сепараторах и большом числе блоков в случае отсева с помощью релаксации большого числа подзадач ЛЭА работает быстрее.

Цель эксперимента:
изучить версию приближенного ЛЭА, исследовать скорость и эффективность

Описание эксперимента:

- Величина сепаратора не превышала 14 переменных
- В качестве декомпозиционного алгоритма используется алгоритм выделения квазиблочной структуры Финкельштейна
- Проведены эксперименты:
  - сравнение точности ЛЭА с оракулом и жадного алгоритма
  - сравнение скорости ЛЭА с оракулом и жадного алгоритма

Модификация 2. ЛЭА с оракулом

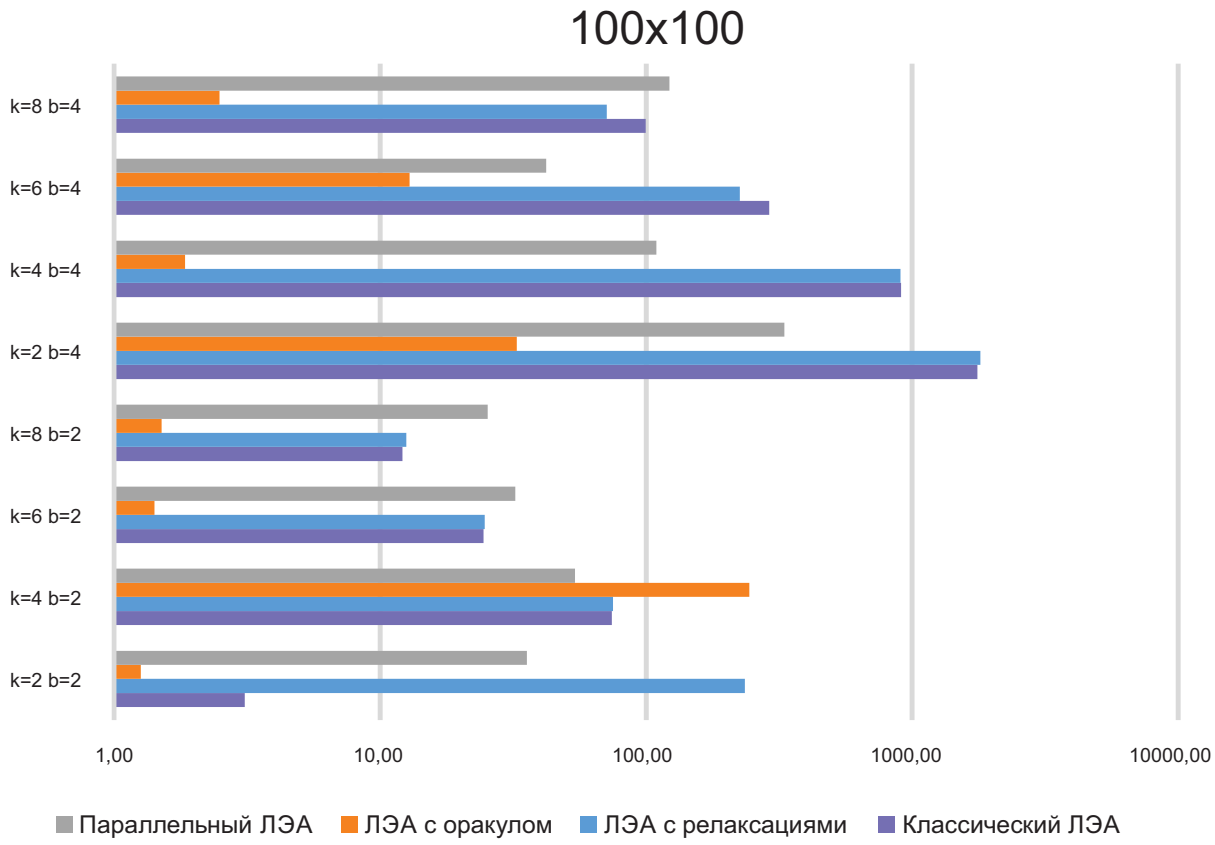
- Алгоритм с оракулом более точный, чем жадный алгоритм
- С увеличением размеров сепараторов приближенный ЛЭА на порядки эффективнее классического ЛЭА

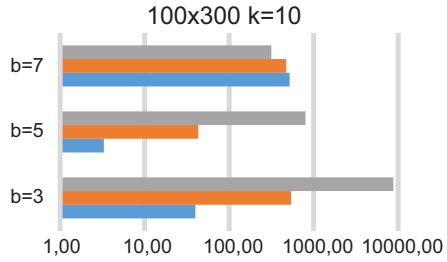
Цель эксперимента:
исследовать, в каких случаях параллельный ЛЭА эффективнее, чем другие модификации ЛЭА

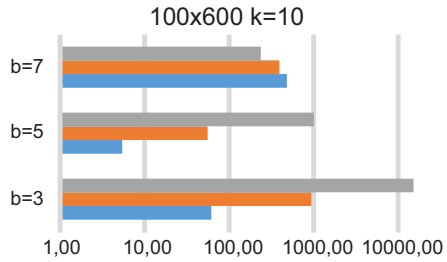
Описание эксперимента:

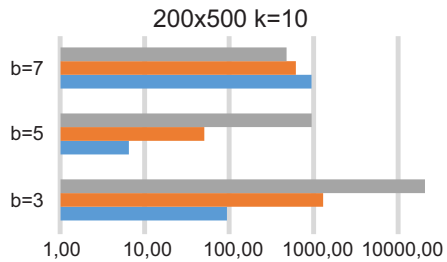
- В качестве декомпозиционного алгоритма используется алгоритм древовидной декомпозиции
- Проведены эксперименты:
  - сравнение с ЛЭА без распараллеливания
  - сравнение с релаксированным ЛЭА
  - сравнение с приближённым ЛЭА

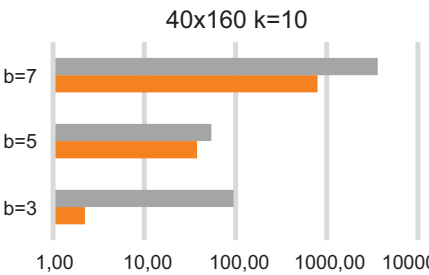
Модификация 3. Параллельный ЛЭА

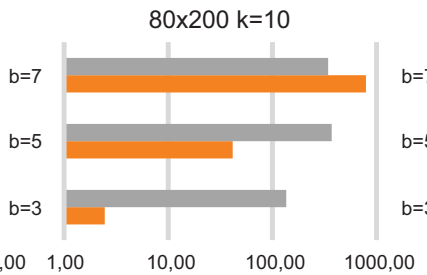


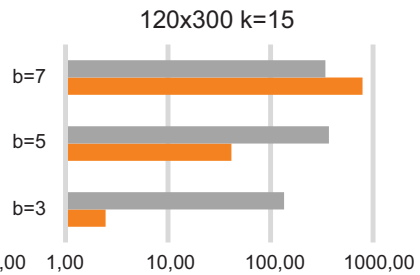


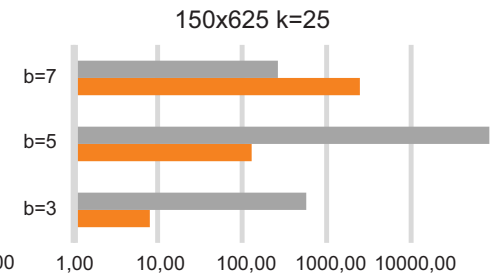












Результаты:

- Параллельный алгоритм неэффективен для решения задач с блоками малой размерности и с небольшими сепараторами.
- При решении задач с большими сепараторами параллельный алгоритм эффективнее, чем другие модификации ЛЭА
- Параллельный ЛЭА позволяет точно решать задачи больших размерностей, для которых классический ЛЭА и ЛАЭ с релаксациями не смогут получить решения

Выводы

- Препроцессинг улучшает время решения больших задач, оптимизируя ограничения
- Модификация декомпозиции Финкельштейна позволяет выделять блочно–древовидную структуру задачи
- Для нахождения оптимального порядка элиминации эффективно выделять эвристики (MINFILL, LEX–BFS)
- Постоптимальный анализ выгодно использовать для пакетов однотипных задач
- Для ускорения ЛЭА предлагаются следующие модификации:
  - ЛЭА с релаксациями решает задачи размерности выше чем 100 на 100 в отличие от классического алгоритма и эффективен для небольших задач
  - ЛЭА с оракулом хорошо использовать в задачах, где необязательно точное решение
  - Параллельный ЛЭА эффективен для задач с большими сепараторами