ИЗВЕСТИЯ РАН. ТЕОРИЯ И СИСТЕМЫ УПРАВЛЕНИЯ, 2019, № 4, с. 74–82

КОМПЬЮТЕРНЫЕ МЕТОДЫ

УДК 004.89, 519.85

ЗАДАЧИ БОЛЬШОЙ РАЗМЕРНОСТИ С КВАЗИБЛОЧНЫМИ МАТРИЦАМИ[1]

© 2019 г. Д. В. Лемтюжникова[a,*], В. Ю. Леонов[b]

[a] *ИПУ РАН, МАИ, Москва, Россия*

[b] *ФИЦ ИУ РАН, Москва, Россия*

**e-mail: darabbt@gmail.com*

Поступила в редакцию 20.02.2019 г.
После доработки 11.03.2019 г.
Принята к публикации 25.03.2019 г.

Рассматриваются разреженные матрицы большой размерности с блочно-лестничной и с блочно-древовидной структурами. Они называются квазиблочными и состоят из независимых блоков, которые связаны попарно друг с другом или еще в более общем виде. Устанавливается зависимость параметров таких матриц, а именно количество ненулевых элементов, число блоков, размерность самих матриц. Также описываются задачи целочисленного программирования с большими матрицами квазиблочной структуры. Используется локальный элиминационный алгоритм для их эффективного решения. Метод представляет собой итеративный процесс, где на каждом шагу исключаются переменные. Изучаются вопросы оптимального порядка исключения. Эта проблема оказывается экспоненциально сложной, что устанавливается с помощью графовой интерпретации понятия блочно-древовидной и блочно-лестничной структур. Рассматриваются вопросы сложности локального элиминационного алгоритма. Это важно, когда возникает вопрос, что лучше, использовать этот метод или применять другие подходы. Представлены результаты численных тестирований, в частности эффективные процедуры оптимальных порядков элиминации. Особое место занимает распараллеливание на компьютерной GRID-системе конкретных квазиблочных задач булевого программирования, которые ввиду больших размерностей не могут быть решены на одном процессоре.

DOI: 10.1134/S0002338819040097

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). Это касается таких известных постановок: размещение объектов, планирование ресурсов, маршрутизация, логистика, 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. 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 [1–3]. Развитие информационных технологий, появление многопроцессорных комплексов, суперкомпьютеров создали условия для разработки алгоритмов ДО с распараллеливанием вычислений.

Эффективными алгоритмами для решения разреженных задач ДО являются локальные элиминационные алгоритмы (ЛЭА). ЛЭА объединяют локальные алгоритмы декомпозиции, алгоритмы несериального динамического программирования, а также алгоритмы сегментной элиминации. Распараллеливание вычислительного процесса ЛЭА может существенно ускорить ре-

[1] Работа выполнена при финансовой поддержке РФФИ (грант № 18-31-00458).

шение задач ДО большой размерности. С развитием вычислительной техники более актуальным становится вопрос сокращения вычислений. Для решения разреженных задач ДО естественным образом выбирается алгоритм, использующий локальные области соответствующей матрицы. Приводятся результаты вычислительного эксперимента, где с помощью ЛЭА решались различные задачи ДО. Представлены результаты исследования эффективности локального алгоритма для решения квазиблочных задач. Также 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 квазиблочных структур, но локальные алгоритмы будут эффективными для задач с ограниченной связностью блоков.

Важно выявление закономерностей в больших данных, в качестве которых выступают разреженные матрицы. При этом повышается эффективность алгоритма для решения задач, соответствующих матрицам; выделение класса задач, для которых применим метод, его ускорение, а также возможности решения задач большой размерности путем распараллеливания. Ниже сформулирована теорема, устанавливающая связь между параметрами матрицы и соответствующей квазиблочной структурой. Также исследованы и реализованы методы выделения квазиблочной структуры для разреженных матриц. Представлен алгоритм перестановки строк и столбцов в матрице для поиска квазиблочных структур в разреженных матрицах. Введены понятия и доказаны свойства графовых структур, соответствующих порядку элиминации. Данные теоремы дают основу доказательства важных свойств в проблеме нахождения оптимального исключения переменных. Протестировано влияние порядка элиминации на скорость ЛЭА. Предложен и реализован ряд модификаций ЛЭА для разреженных задач ДО с квазиблочной структурой. Реализовано распараллеливание задач с квазиблочной структурой на GRID. Представленный обзорный материал является презентацией недавно вышедшей в свет монографии [4]. Другая (оригинальная) часть представляет собой алгоритмы выделения квазиблочных структур для многомерных матриц. В качестве основного рассматривается алгоритм Ю.Ю. Финкельштейна [5], основанный на перестановке строк и столбцов для формирования блочно-лестничной (БЛ) структуры. Установлены недостатки подхода: они связаны с большим числом связывающих переменных между блоками, а также пустыми блоками, из-за чего применение таких структур затруднено. Реализованы модификации основного алгоритма, при помощи которых можно получить оптимальную структуру, пригодную для использования в решении практических задач. Приводится сравнение основного и модифицированного алгоритмов относительно параметров полученных структур, таких, как количество блоков и связывающих переменных между блоками. Установлена эффективность модификации. Описывается модификация основного алгоритма, применяемая для выделения блочно-древовидных (БД) структур.

1. Квазиблочная структура в разреженных матрицах. Матрица является разреженной, если содержит преимущественно нулевые элементы. Интерес к разреженности заключается в экономии на вычислениях. Зная структуру матрицы, можно разделять соответствующую задачу на подзадачи. Другими словами, информация о том, как упорядочены ненулевые элементы в матрице, позволяет произвести декомпозицию матрицы на блоки преимущественно с ненулевыми элементами и осуществлять решение задач в блоках независимо друг от друга.

Здесь естественным образом возникает проблема хранения разреженных матриц. Для хранения используются разные, например, ленточные представления. Выделяют полные и неполные схемы в зависимости от того, представлена вся матрица или только ее часть. Также выделяют упорядоченные и неупорядоченные схемы в зависимости от того, произволен порядок хранения элементов или упорядочен.

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. Обзор литературы по разреженным матрицам представлен в [6].

Классифицируем разреженные матрицы. Очень широкими будем называть матрицы, в которых число столбцов превышает число строк в 2 раза и более. Широкие матрицы определим как

матрицы, в которых число столбцов превышает число строк менее чем в 2 раза. В квадратных матрицах число столбцов равно числу строк, а в узких – число строк превышает число столбцов. 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. Другими словами, многие подзадачи, соответствующие таким матрицам, могут быть решены независимо друг от друга, что существенно экономит вычислительное время. В частности, такой структурой является БД-структура.

О п р е д е л е н и е 1. Матрица A, в которой n столбцов, m строк и z ненулевых элементов, таких что выполняется $0.5mn > z$, называется *разреженной*.

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

О п р е д е л е н и е 2. Столбец j матрицы A, в котором существует хотя бы пара ненулевых элементов a_{ij} и $b_{i',j}$ в строках i и $i' \neq i$, будем называть *связывающим*. Взаимосвязью двух столбцов j_1 и j_2 в матрице A называется единовременное вхождение ненулевых элементов этих столбцов в строку i.

Перейдем к определению графа взаимосвязей.

О п р е д е л е н и е 3. *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.

Подразумевается, что индексы вершин графа взаимосвязей и номера столбцов матрицы A эквивалентны. Введем понятие окрестности для столбца матрицы A в графовом представлении.

О п р е д е л е н и е 4. Множество вершин, т.е. связанных ребром с вершиной x в графе взаимосвязей $G(X, E)$, обозначается Ω и называется *окрестностью* вершины x.

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.

Перейдем к определению квазиблочных структур матрицы. Определим граф пересечения окрестностей [4]. Представим граф $G(X, E)$ в виде системы окрестностей $\Omega_1 = (S_1, U_1)$, $\Omega_2 = (S_2, U_2)$, ..., $\Omega_k = (S_k, U_k)$ 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, $r = 1, \ldots, k$.

О п р е д е л е н и е 5. *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 $S_{r_1} \cap S_{r_2} \neq \varnothing$.

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

С в о й с т в о 1. Объединение множеств номеров строк для каждой окрестности соответствует множеству номеров строк всей матрицы:

$$\bigcup_{r=1}^{k} U_r = M = \{1, \ldots, m\}.$$

С в о й с т в о 2. Объединение множеств номеров столбцов для каждой окрестности соответствует множеству номеров столбцов всей матрицы

$$\bigcup_{r=1}^{k} S_r = N = \{1, \ldots, n\}.$$

С в о й с т в о 3. Множества номеров столбцов, соответствующие любым двум окрестностям из заданной системы окрестностей, не пересекаются:

$$U_{r_1} \cap U_{r_2} = \varnothing, \quad r_1 \neq r_2.$$

Свойство 4. Множества номеров строк, соответствующие любым трем окрестностям из заданной системы окрестностей, не пересекаются одновременно:

$$S_{r_1} \cap S_{r_2} \cap S_{r_3} = \varnothing, \quad \forall r_1, r_2, r_3.$$

Перейдем к определению БД-структуры матрицы A.

Определение 6. Граф G_Ω, для которого выполняются свойства 1–4 и при этом он является деревом, называется БД-структурой. Матрицу, соответствующую БД-структуре будем называть *блочно-древовидной*, выделенную вершину ν_r – *корнем*.

БД-структура определяется соотношением “предок–потомок”. При этом если (ν_r, ν_1), ..., (ν_{p-1}, ν_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 7. Граф G_Ω, для которого выполняются свойства 1–4, и при этом он является цепью, называется БЛ-структурой. Матрицу, соответствующую БЛ-структуре, будем называть *блочно-лестничной*.

Определим компоненты БД-структуры. Введем определение блока, степени БД-структуры и сепаратора.

Определение 8. Вершины ν_{r_i}, соответствующие каждой r-й окрестности, будем называть *блоками* БД-структуры, количество блоков обозначим k.

Определение 9. *Степенью* БД-структуры будем называть $\rho = \max\{\rho_1, ..., \rho_k\}$, где $\rho_1, ..., \rho_k$ – степени каждого блока в БД-структуре.

Определение 10. Для данного связного графа $G(X, E)$ и множества индексов вершин $S \subseteq X$, таких что подграф $G[X\backslash S]$ графа $G(X, E)$ несвязан, S называется *сепаратором*.

В [4] доказана взаимосвязь между компонентами БД-структуры и параметрами разреженной матрицы с помощью следующей теоремы.

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

При этом параметры n, m, z связаны следующим образом:

для очень широких матриц $m \le 4$, $n \le 2m$, $2n - m + 1 \le z < 0.5mn$;

для широких матриц:

а) $4 \le m \le 6$, $m + 1 \le n \le 7$, $n + m - 1 \le z < 0.5mn$,

б) $5 \le m \le 7$, $8 \le n \le 2m - 1$, $n + m - 1 \le z < 0.5mn$,

в) $m \ge 8$, $m + 1 \le n \le 2m - 1$, $n + m - 1 \le z < 0.5mn$;

для квадратных матриц $m = n$, $n \ge 4$, $2n - 1 \le z < 0.5n^2$;

для узких матриц:

а) $6 \le m \le 7$, $5 \le n \le m - 1$, $m + n - 1 \le z < 0.5mn$,

б) $m \ge 8$, $5 \le n \le 7$, $m + n - 1 \le z < 0.5mn$,

в) $m \ge 9$, $8 \le n \le m - 1$, $m + n - 1 \le z < 0.5mn$.

2. Алгоритмы выделения квазиблочных структур. Основные конструкции выглядят так. Пусть A_{mxn} – разреженная матрица, p – индекс вершины x_p, $p \in \{1, 2, ..., n\}$. Далее необходимо выделить окрестности вершины x_p последовательно возрастающих порядков. 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:

$$U_r = \begin{cases} U'_1(x_p), & r = 1, \\ U'_r(x_p)\backslash U'_{r-1}(x_p), & 1 < r < k, \\ U'_k(x_p)\bigcup_{k+1}(x_p)\backslash U'_{k-1}(x_p), & r = k. \end{cases}$$

Затем находятся следующие множества индексов столбцов:

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

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. Данный алгоритм содержит следующий недостаток. Сепараторы могут оказаться очень большими, т.е. состоять из большого числа переменных. Это крайне усложняет работу с матрицей (например, используя ЛЭА). Поэтому есть смысл добавлять ограничение на размер сепаратора и, если он превышает допустимый, выполнять процедуру слияния блоков. Другими словами, если сепаратор 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 $U_a \cup U_b$. Сепараторами этого блока будут множества $S_{a\,-\,1,\,a}$ и $S_{b,\,b\,+\,1}$, а свободными переменными – $S_{new} = S_a \cup \cup S_b$.

Реализован эксперимент сравнения основного алгоритма и его модификации. Тестовые задачи генерировались на основе уже существующих графов из библиотеки CSP, которая содержит различные классы графов для задач анализа данных, например из NASA и ISCAS. Были выбраны две группы: “дубликатор” и “крендель” согласно различиям в графовых структурах. Для построения ограничения бралось ребро графа, задавалось соответствующее множество переменных. Затем случайным образом определялись коэффициенты при переменных. Целевая функция строилась по всем вершинам графа, коэффициенты при переменных также определялись случайным образом. Для каждого графа применялись описанные выше алгоритмы упорядочивания переменных. Затем задачи с заданными порядками решались с помощью ЛЭА. Алгоритм для решения подзадач реализован на онлайн-платформе алгебраического языка моделирования AMPL (www.ampl.com), в качестве решателя выбран SCIP (solving constraint integer programs) – решатель задач целочисленного программирования. Он использует технологию ветвления задачи на более мелкие подзадачи и различные релаксации, в частности методы отсечения, а также дает возможность использовать произвольные ограничения. Кроме того, в [7] описан механизм, позволяющий ему решать выпуклые и невыпуклые задачи смешанного целочисленного программирования.

Рассмотрим некоторые результаты для тестовых задач. Класс “дубликатор” описывает сгенерированные случайным образом задачи не удовлетворяющие ограничениям. Здесь можно отметить, что существенным образом сократилось количество блоков и максимальный сепаратор значительно уменьшился. Класс “крендель” описывает граф, раскрашенный в два цвета, который построен таким образом, чтобы не удовлетворять ограничениям. Для задач этого класса существенно уменьшилось количество подзадач, которые потребуется решить.

3. Локальный алгоритм и порядок исключения переменных. Под задачей дискретной оптимизации подразумевается задача следующего вида:

$$z = \mathrm{extr}_{x \in G} f(x),$$

G – конечное или счетное множество допустимых решений, а extr – экстремум функции, т.е. ее минимум или максимум. При этом x будем называть вектором решений, а $f(x)$ – целевой функцией задачи. Задачей целочисленного линейного программирования (ЦЛП) будет называться такая задача, где $f(x)$ линейна, а вектор решений принимает целочисленные значения. Обзор литературы по декомпозиционным методам решения блочных оптимизационных задач показан в [8].

Задачи дискретной оптимизации большой размерности, встречающиеся на практике, часто имеют разреженную структуру. Для решения таких задач эффективно использовать локальные методы Журавлева – класс комбинаторных методов, которые исследуют окрестности переменных, решают соответствующие окрестностям подзадачи и на основе полученных решений восстанавливают глобальное решение задачи. Выделение БД-структуры для разреженных матриц позволяют декомпозировать задачу и решать подзадачи для отдельных блоков независимо друг от друга. Для этого процесса необходимо понимать, каким образом связан ход алгоритма с декомпозицией, чтобы знать, может ли он использоваться для непосредственного выделения БД-структуры в матрице за счет внутренней организации процессов. Такие алгоритмы будем называть локальными алгоритмами, а встроенные в них алгоритмы для решения задач в блоках –

внутренними. В теории локальных алгоритмов Журавлева выделены две основные теоремы – единственности и мажорантности [9]. 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. Другими словами, в 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. Значит, от того, в каком порядке мы элиминируем переменные или подмножества переменных, зависит скорость нахождения оптимального решения. Доказательство этой теоремы имеет неконструктивный характер, т.е. не позволяет осуществить прямое построение эффективного наилучшего алгоритма. В [10] сформулирован критерий существования оптимального порядка элиминации, который заключается в следующем.

Т е о р е м а 2. Для графа взаимосвязей существует оптимальный порядок элиминации тогда и только тогда, когда можно построить дерево декомпозиции, каждая вершина которого является кликой в соответствующем графе ограничений.

Порядок элиминации α – это выбор порядка решения подзадач. Теорема 2 определяет, в каком случае существует такой порядок решения подзадач, при котором работа ЛЭА будет оптимальной. Задача поиска оптимального упорядочения является NP-полной [11], поэтому на практике для нахождения элиминационной последовательности переменных используются всевозможные эвристики. Для задач оптимизации со специальной структурой эвристики выступают как алгоритмы упорядочивания суперпеременных для последующей элиминации.

Далее будем рассматривать класс локальных методов – ЛЭА решения задач ДО, использующие окрестности переменных [10]. Важной особенностью ЛЭА является вычисление именно локальной информации (т.е. информации об элементах окрестностей элементов) при решении разреженных дискретных задач, поэтому ЛЭА позволяют осуществлять вычисление глобальной информации о решении всей задачи с помощью локальных вычислений (обычно решений подзадач). Основная идея ЛЭА состоит в последовательном исключении переменных с сохранением информации об этих переменных. Процедура ЛЭА разбивается на две части. Первая часть состоит в элиминации элементов, вычислении и запоминании информации в виде локальных решений и получении в конце значения критерия. Вторая часть – это поиск глобального решения всей задачи по найденным в прямой части таблицам с локальными решениями, обеспечивающими в прямой части достижение критерия. Первая часть ЛЭА состоит из нескольких этапов: определение порядка исключения переменных, согласно некоторой эвристике, запись локальной информации об этих переменных в виде новых зависимостей, добавляемых к задаче, а также исключение просмотренных элементов и использованных зависимостей, согласно порядку исключения переменных.

4. Сложность и эффективность ЛЭА. При использовании локальных алгоритмов для решения задач с БД-структурой возникают следующие вопросы. Во-первых, всегда ли возможно избежать полный перебор при решении таких задач? Если это возможно не всегда, представляет интерес получение соответствующих достаточных условий, позволяющих исключить полный перебор. Во-вторых, всегда ли использование локального алгоритма в сочетании с некоторым алгоритмом для решения задач в блоках эффективнее применения упомянутого алгоритма самого по себе? В-третьих, есть ли смысл рассматривать БД-структуры или их следует свести к другим, более простым? В связи с этим представляет интерес сравнение оценок эффективности алгоритмов в сочетании с некоторым алгоритмом, с помощью которого решаются задачи внутри блоков для решения некоторой задачи с БД-структурой и оценка эффективности алгоритма, использованного для решения всей задачи, так что в первом случае учитывается БД-структура задачи, а во втором – нет. Оценки эффективности ЛЭА при решении задач с БД-структурой характеризуются деревом пересечения окрестностей D с весами вершин n_r и весами ребер $n_{rr'}$, с n булевыми переменными и k блоками:

$$E_{ЛЭА}(n,k,D) = \sum_{r=1}^{k} 2^{N_r} \varphi_{\text{Ъ}}(n_r),$$

где $\varphi_{\text{Э}}$ – оценка эффективности алгоритма, решающего задачи в блоках B_r, а N_r определяется следующим образом:

$$N_r = n_{p,r} + \sum_{r' \in J_r} n_{rr'}.$$

Напомним, что под вычислительной сложностью алгоритма понимают количество условных шагов или операций, необходимых для решения задачи. При введении оценок сложности алгоритмов решения комбинаторных задач обычно различаются индивидуальная и массовая задачи (множество индивидуальных задач). При этом для оценки трудоемкости алгоритмов используются следующие характеристики: временная вычислительная сложность алгоритма – время, затрачиваемое алгоритмом для решения задачи, емкостная сложность алгоритма – объем памяти, необходимый для реализации алгоритма. Для сглаживания резких различий в поведении алгоритма при переходе от одной индивидуальной задачи к другой можно рассматривать все индивидуальные задачи одной размерности n вместе и определить сложность алгоритма для этой размерности задачи как число шагов (или условных операций) алгоритма в худшем случае, т. е. находится верхняя оценка сложности алгоритма, говорящая о том, что данный алгоритм решает задачу не более чем за $T(n)$ условных единиц времени. Если для некоторого алгоритма верхняя оценка сложности меньше, чем для других алгоритмов, то говорят, что этот алгоритм имеет характеристику в худшем случае лучше, чем другие алгоритмы.

Ориентация на худший случай иногда приводит к пессимистическим прогнозам поведения алгоритмов, так, хорошо известный симплекс-алгоритм на практике работает как полиномиальный алгоритм (это же показывает и теоретический анализ его поведения в среднем), хотя оценка сложности в худшем случае для него является экспоненциальной.

В связи с этим с практической точки зрения часто больший интерес представляет средняя оценка вычислительной сложности, позволяющая судить о поведении алгоритма в среднем на некотором классе задач, однако средняя оценка вычислительной сложности тоже не дает полной характеристики эффективности алгоритма, так как возможны задачи, при решении которых сложность алгоритма превысит эту оценку.

Важной характеристикой алгоритма является асимптотическая оценка сложности – порядок скорости роста сложности алгоритма при увеличении размерности задачи. Асимптотические оценки вычислительной сложности алгоритмов позволяют судить об их поведении при решении задач большой размерности и определить границы применимости алгоритма. Асимптотическая средняя оценка эффективности $E(\varphi, n, k, \tilde{n})$ ЛЭА в сочетании с алгоритмом ДО, имеющим оценку эффективности $\varphi(n)$, при решении БД-задач ДО с ограниченной связностью при $n \to \infty$ имеет вид [12]:

$$E(\varphi, n, k, \tilde{n}) \asymp \begin{cases} \dfrac{2^{n-k+1}}{n^{k-1}}, & \varphi(n) = 2^n, \\ \dfrac{2^{n-k+1}}{n^{\alpha+k-1}}, & \varphi(n) = \dfrac{2^n}{n^\alpha}, \\ n^s, & \varphi(n) = n^s. \end{cases}$$

Таким образом, ЛЭА достаточно эффективен даже для решения БД-задач ДО с небольшой связностью, причем его оценка эффективности существенно зависит от алгоритма, с помощью которого решаются подзадачи на каждом шагу алгоритма.

В [13] приведены эксперименты по исследованию эффективности ЛЭА. Для решения задач применялся решатель SYMPHONY и ЛЭА в сочетании с данным решателем. В результате было установлено что использование алгоритма в сочетании с ЛЭА гораздо эффективнее. Также было замечено, что при малых размерностях эффективность ЛЭА отсутствует.

5. Приближенные и параллельные алгоритмы. Приближенные методы широко применяются при решении задач ЦЛП, 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 реше-

ния задачи. Примером эвристического алгоритма может быть алгоритм решения задачи коммивояжера, в котором на каждом шагу реализуется переход в ближайшую из оставшихся точку. Алгоритмы такого типа носят название “greedy” (“жадные” алгоритмы). Эти алгоритмы 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 применять несколько алгоритмов этого типа, изменяя правила выбора окрестности.

ЛЭА 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.

Трудности с решением достаточно больших подзадач ЛЭА 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. Основным недостатком ЛЭА является полный перебор по множествам сепараторов, что дает возможность большого объема перебора при больших размерах сепаратора. Преодолеть этот недостаток и способствовать успешному решению практически важных задач ЦЛП специальной структуры с большими размерами сепараторов может введение процедуры, позволяющей “предсказать” искомые оптимальные (или близкие к оптимальным) значения перемычек. Зная значения перемычек, задачи ЦЛП, соответствующие блокам, можно решать отдельно, независимо друг от друга. Очевидно, что возможность узнать каким-то образом оптимальные значения представляется сомнительной. Тем не менее можно найти значения перемычек, близкие к оптимальным, с помощью замены соответствующих блоков задач ЦЛП на их релаксации, а затем решить построенные задачи ЦЛП с помощью ЛЭА. Таким образом все возможные наборы перемычек перебираются полностью, однако трудоемкость выполнения каждого шага существенно снижена за счет решения не исходных, а релаксированных и поэтому более легко решаемых подзадач. Если все же и с релаксированными подзадачами рассматриваемая задача ЦЛП не поддается точному решению, можно применять случайный выбор значений перемычек. Кроме того, можно использовать неполный перебор перемычек.

Разработка параллельного программного обеспечения для научных целей – отдельная проблема. Выбор целевой вычислительной платформы для реализации ЛЭА может существенно влиять на используемый способ распараллеливания. Таким образом, развитие параллельного программного обеспечения часто требует большей привязанности к времени и энергии, чем разработки итеративных аналогов. В последнее десятилетие наблюдается повышенный интерес к GRID-вычислениям. 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 для большого числа пользователей. Создание вычислительных сетей позволяет пользователям задействовать простое объединение сотен тысяч подключенных компьютеров и обеспечить необходимую вычислительную мощность. GRID-вычисления представляют собой совокупность распределенных гетерогенных ресурсов, которые могут применяться в качестве группы выполняемых крупномасштабных приложений. GRID-вычисления используют совокупность слабосвязанных разнородных вычислительных ресурсов как единый объект. Реализовано распараллеливание задач с квазиблочной структурой. При этом осуществляется независимое решение промежуточных блочных задач на отдельных процессорах. Приводятся конкретные задачи булевого программирования с матрицами БЛ-/БД-структуры, которые успешно решаются на GRID-системе, хотя простое решение таких задач на одном процессоре было невозможно, а получить решение на GRID без ЛЭА достаточно затратно по времени [14]. Задача булевого программирования с БД-структурой имеет 15 блоков, в каждом из которых порядка 50000 переменных и 100 ограничений. Количество сепараторов равно 4. Алгоритм распараллеливания работает 17 мин. Тест в БЛ-структуре содержит 5 блоков с количеством переменных и ограничений – 100000 и 100. Количество сепараторов равно 4. Время решения составляет 6 ч.

Заключение. Эффективность локального алгоритма теоретически и экспериментально исследована недостаточно полно, поэтому остается актуальным поиск удачных модификаций ЛЭА и его сочетаний с различными точными и приближенными решателями ДО. В ранее приведенных конструкциях фигурировали линейные задачи. Однако представляет интерес обобщение на нелинейные и другие постановки, например, по аналогии, как это делается в [15, 16].

СПИСОК ЛИТЕРАТУРЫ

1. *Цурков В.И.* Декомпозиция задач большой размерности. М.: Наука, 1981. 352 с.
2. *Авербах И.Л., Цурков В.И.* Оптимизация в больших задачах с целочисленными переменными. М.: Наука, Физматлит, 1995. 288 с.
3. *Цурков В.И.* Принцип декомпозиции для блочно-сепарабельных систем // Докл. АН СССР. 1979. Т. 246. № 1. С. 27–31.
4. *Щербина О.А., Лемтюжникова Д.В., Цурков В.И.* Многомерные задачи с квазиблочными матрицами. М.: Физматлит, 2018. 256 с.
5. *Финкельштейн Ю.Ю.* Приближенные методы и прикладные задачи дискретного программирования. М.: Наука, 1976. 265 с.
6. *Ковков Д.В., Лемтюжникова Д.В.* Декомпозиция в многомерных задачах с разреженными матрицами // Изв. РАН. ТиСУ. 2018. № 1. С. 98–110.
7. *Vigerske S., Gleixner A.* SCIP: Global Optimization of Mixed-integer Nonlinear Programs in a Branch-and-cut Framework // Optimization Methods and Software. 2018. V. 33. № 3. P. 563–593.
8. *Лемтюжникова Д.В., Ковков Д.В.* Задачи дискретной оптимизации с квазиблочными матрицами // International J. Open Information Technologies. 2017. № 10. С. 1–8.
9. *Журавлев Ю.И.* Избранные научные труды. М.: Магистр, 1998. 420 с.
10. *Щербина О.А.* Локальные элиминационные алгоритмы для разреженных задач дискретной оптимизации: дис. ... докт. физ.-мат. наук. М.: ВЦ РАН, 2010.
11. *Yannakakis M.* Computing the Minimum Fill-in is NP-complete // SIAM J. Alg. Disc. Meth. 1981. V. 2. P. 77–79.
12. *Щербина О.А.* Асимптотические оценки эффективности локальных алгоритмов в дискретном программировании // ЖВМФ и МФ. 1982. Т. 22. № 6. С. 1360–1368.
13. *Лемтюжникова Д.В., Ковков Д.В.* Тестирование алгоритмов для целочисленных квазиблочных задач оптимизации // Вестн. МГТУ им. Н.Э. Баумана. Сер. Информационные технологии. 2018. № 1. С. 59–75.
14. *Волошинов В.В., Лемтюжникова Д.В., Цурков В.И.* Распараллеливание на grid задач дискретной оптимизации с матрицами квазиблочной структуры // Изв. РАН. ТиСУ. 2017. № 6. С. 35–40.
15. *Миронов А.А., Федорчук В.В., Цурков В.И.* Минимакс в транспортных моделях с целочисленными ограничениями: II // Изв. РАН. ТиСУ. 2005. № 5. С. 732–752.
16. *Миронов А.А., Федорчук В.В., Цурков В.И.* Минимакс в транспортных моделях с целочисленными ограничениями: I // Изв. РАН. ТиСУ. 2003. № 4. С. 562–574.