Динамические системы, вып. 24 (2008), 83–98

ДИНАМИЧЕСКИЕ СИСТЕМЫ
Межведомственный научный сборник

УДК 519.68

Роль графовых структур в теории локальных элиминационных алгоритмов[1]

О.А. Щербина
University of Vienna,
Vienna 1090, Austria. *E-mail: oleg.shcherbina@univie.ac.at*

Аннотация. Рассмотрено использование различных графовых структур в теории локальных элиминационных алгоритмов (ЛЭА) для решения дискретных задач, в том числе элиминационных графов и элиминационных деревьев. Показано, что бесконтурный орграф вычислительной процедуры ЛЭА является элиминационным деревом. Обсуждаются возможности построения древовидной декомпозиции на основе элиминационного дерева, а также представление элиминационного процесса с помощью множеств достижимости.

1. Введение

Для решения ряда разреженных дискретных задач, возникающих в самых разных областях приложений (в том числе задач искусственного интеллекта и дискретной оптимизации [8]), содержащих огромное число переменных и/или ограничений, что создает сложности при попытке решения этих задач с помощью современных решателей, могут быть использованы графовые декомпозиционные методы [26]. Возросший интерес к этим методам обусловлен результатами Courcelle [22], Arnborg et al. [17], доказавших, что ряд NP-трудных задач, поставленных в монадической логике второго порядка, могут быть решены за полиномиальное время с помощью методов динамического программирования на графах (описывающих структуру задачи) с ограниченной древовидной шириной.

К графовым декомпозиционным подходам относится класс локальных элиминационных алгоритмов (ЛЭА) вычисления информации [11], включающий локальные алгоритмы декомпозиции [4], [9], алгоритмы несериального динамического программирования (НСДП) ([19], [10], алгоритмы сегментной элиминации [23], методы древовидной декомпозиции [12], [26].

Применение ЛЭА для решения задач дискретной оптимизации (ДО) рассмотрено в [10]. В [13] показано, что ЛЭА может быть использован и для решения неоптимизационных задач (задач удовлетворения ограничений), которые можно

[1]Работа выполнена при финансовой поддержке австрийского фонда FWF, грант No. P17948-N13

разбить на подзадачи и использовать полученные решения меньших подзадач при решении больших. Связь НСДП с локальными алгоритмами ([4], [9], [10]) обусловлена тем, что, как и локальные алгоритмы, НСДП решает задачи, преобразуя локальную информацию в глобальную. Одной из общих для этих методов графовых интерпретаций является **элиминационная игра** (Parter [28]).

В [15] предложен общий класс локальных элиминационных алгоритмов вычисления информации для решения дискретных разреженных задач, позволяющих осуществлять вычисление **глобальной информации** о решении задачи с помощью **локальных вычислений** на основе анализа окрестностей элементов задачи.

В настоящей статье анализируются возможности графовых структур в теории локальных элиминационных алгоритмов, показано, что бесконтурным орграфом вычислительной процедуры ЛЭА является **элиминационное дерево**, предложенное ранее в работе [27] для описания процесса элиминации переменных при решении разреженных систем линейных уравнений.

1.1. Основные понятия

Рассмотрим задачу ДО

$$F(x_1, x_2, \ldots, x_n) = \sum_{k \in K} f_k(Y^k) \to \max \tag{1.1}$$

при ограничениях

$$g_i(X^i)\ R_i\ 0,\ i \in M = \{1, 2, \ldots, m\}, \tag{1.2}$$

$$x_j \in D_j,\ j \in \{1, \ldots, n\}, \tag{1.3}$$

где

$$Y^k \subseteq \{x_1, x_2, \ldots, x_n\},\ k \in K = \{1, 2, \ldots, t\}; \tag{1.4}$$

$$X^i \subseteq X = \{x_1, x_2, \ldots, x_n\},\ R_i \in \{\leq, =, \geq\},\ i \in M = \{1, 2, \ldots, m\}; \tag{1.5}$$

$$D_j\text{— дискретное множество значений переменной } x_j,\ j = 1, \ldots, n. \tag{1.6}$$

Определение 1. ([19], [10]) Две переменные x и y взаимосвязаны в задаче ДО с ограничениями, если они появляются вместе в одном компоненте целевой функции (ЦФ) или в одном и том же ограничении (другими словами, если переменные $x,\ y$ входят одновременно во множество X^i или во множество Y^k).

Определение 2. Графом взаимосвязей [19] задачи ДО называется неориентированный граф $G = (X, E)$, для которого

1. множество вершин X соответствует множеству переменных X задачи ДО;
2. две вершины $x,\ y$ графа G соединены ребром $(x,\ y)$ тогда и только тогда, когда соответствующие им переменные взаимосвязаны.

В дальнейшем будем отождествлять переменные задачи ДО с вершинами графа взаимосвязей. Ниже под графом будем подразумевать неориентированный граф, если не оговорено другое.

Ориентированный граф (орграф) [3] будем обозначать ниже в виде $\overrightarrow{G} = (X, \overrightarrow{E})$, где X — конечное множеств вершин, $\overrightarrow{E}$ — множество дуг (ориентированных ребер), $\overrightarrow{E} \subseteq X^2$.

Определение 3. Стягивающее дерево — для данного неориентированного графа суграф (т.е. часть графа, имеющая то же множество вершин, что и сам граф [2]) в виде дерева.

Определение 4. Корневое дерево — это дерево $T = (X, E)$ с выделенной вершиной r, называемой **корнем** T.

Корневое дерево полностью описывается отношением «предок–потомок». При этом, если $(r, s), \ldots, (x, y)$ — путь от корня r в вершину y , то x называют **предком** вершины y.

Основным понятием в теории локальных алгоритмов является понятие **близости элементов**, определяемое понятием **окрестности**.

Определение 5. Две вершины x и y в графе $G = (X, E)$ называются **соседними** или **смежными**, если $(x, y) \in E$.

Определение 6. ([3]) **Клика** - подмножество вершин X' графа G, в котором любые две вершины смежны, т.е. порожденный им подграф $G(X')$ является полным.

Определение 7. Множество вершин, соседних с вершиной $x \in X$ в графе $G = (X, E)$, обозначается $Nb_G(x)$ (или $Nb(x)$) и называется **окрестностью** вершины x.

Определение 8. ([1]) **Транзитивное замыкание орграфа** (Transitive closure of a directed graph) — для данного орграфа $\overrightarrow{G}$ орграф $\overrightarrow{G}^*$ с тем же множеством вершин, что и $\overrightarrow{G}$, в котором дуга (v, w) существует тогда и только тогда, когда w достижима из v в $\overrightarrow{G}$.

Определение 9. ([7]) **Транзитивная редукция орграфа** (Transitive reduction of a directed graph) — для данного орграфа $\overrightarrow{G}$ орграф с наименьшим числом дуг, транзитивное замыкание которого совпадает (изоморфно) с транзитивным замыканием исходного графа $\overrightarrow{G}$.

Таким образом, если в орграфе G имеется ориентированный путь из вершины x в вершину y с длиной, большей 1, то имеющееся ориентированное ребро x, y может быть удалено. Удаление всех таких ребер из орграфа позволяет получить его транзитивную редукцию.

Поиск в глубину. Идея поиска в глубину (DFS — depth first search) [30] — одного из основных графовых алгоритмов — состоит в следующем. Поиск начинается с некоторой фиксированной вершины v_0. Затем выбирается произвольная вершина u, соседняя с v_0, и повторяется поиск соседней с u вершины. В общем случае предположим, что мы находимся в вершине v. Если существует новая, еще не просмотренная, вершина u, $u \in Nb(v)$, рассматриваем ее (помечаем ее как просмотренную) и, начиная с нее, продолжаем поиск. Если же не существует ни одной новой вершины, соседней с v, то говорим, что вершина v **использована**, возвращаемся в вершину, из которой мы попали в v, и продолжаем процесс (если $v = v_0$, то поиск закончен). Таким образом, поиск в глубину из вершины v основан на поиске в глубину из всех новых вершин, соседних с v.

2. Локальные элиминационные алгоритмы вычисления информации

2.1. О локальных алгоритмах

Использование локальной информации [4], [5] при изучении сложных дискретных систем, при разработке методов декомпозиции для решения больших разреженных дискретных задач является перспективным подходом, причем упомянутые задачи принадлежат как области дискретной оптимизации [6], [8], так и области искусственного интеллекта [23], информатики [18], линейной алгебры (здесь разработаны мультифронтальные методы решения разреженных систем линейных уравнений [29], имеющие декомпозиционный характер).

При изучении сложных объектов не всегда возможно получить (или вычислить) глобальную информацию об объекте в целом, поэтому представляет интерес получение глобальной информации об объекте, используя локальные вычисления для частей объекта.

Локальный алгоритм (ЛА) вычисления информации [4] изучает элементы в порядке, задаваемом алгоритмом упорядочения A_π, используя при этом локальную информацию об элементах из окрестности [5] данного элемента. При этом производится вычисление функции φ, значение которой на каждом шаге алгоритма будет определять вид отметки, выставляемой на этом шаге.

ЛА декомпозиции [9], [10], [11], [12] задач ДО имеют свою специфику, состоящую в том, что они вычисляют не предикаты, а, используя принцип оптимальности Беллмана, находят оптимальные решения подзадач, соответствующих блокам задачи ДО. Шаг ЛА декомпозиции $\mathfrak{A}$ заключается в переходе от окрестности S_p к окрестности S_{p+1}, на каждом шаге алгоритма для каждого фиксированного набора переменных граничного кольца запоминаются значения переменных соответствующей окрестности, причем запоминается информация не о предикатах, а о значениях переменных в решениях подзадач; такую информацию Ю.И. Журавлев предложил называть **индикаторной информацией**.

Важной особенностью локальных алгоритмов является вычисление и использование именно **локальной информации** (т. е. информации об элементах окрестностей [5] элементов) при решении разреженных дискретных задач, поэтому локальные элиминационные алгоритмы (ЛЭА) вычисления информации позволяют осуществлять вычисление глобальной информации о решении всей задачи с помощью локальных вычислений (обычно решений подзадач), используя структуру задачи.

Структура дискретной задачи оптимизации задается структурным графом, и может быть задана как исходными элементами (переменными) задачи с заданием системы окрестностей [5] этих переменных, графа взаимосвязей и порядка просмотра этих переменных с помощью ЛЭА, так и различными производными структурами — блочными [11, 19], блочно-древовидными [12], задаваемыми так называемым структурным конденсированным графом, в котором вершинами являются подмножества переменных задачи.

ЛЭА вычисляет информацию о локальных элементах структуры дискретной задачи оптимизации, задаваемой структурным графом, записывая вычисленную локальную информацию об этих элементах в виде новых компонентов целевой функции, добавляемых к задаче, элиминируя затем просмотренные элементы и использованные ограничения.

Процедура ЛЭА состоит из двух частей:

- **прямая часть** — выделение локальных элементов и их окрестностей в текущем структурном графе и соответствующих им подзадач, вычисление и запоминание локальной информации в виде новых компонентов целевой функции, добавляемых к задаче, а также запоминание таблиц с локальными решениями, элиминация просмотренных локальных элементов и использованных ограничений, соответствующее изменение структурного графа, получение оптимального значения целевой функции;

- **обратная часть** — нахождение глобального решения исходной дискретной задачи оптимизации, обеспечивающего достижение оптимального значения целевой функции по найденным в прямой части таблицам с локальными решениями.

Алгоритмическая схема ЛЭА представляет собой бесконтурный орграф, вершины которого соответствуют локальным подзадачам, а ребра — выражают информационную зависимость подзадач друг от друга. Знание алгоритмической схемы позволяет оценить вычислительную сложность ЛЭА, а также организовать распараллеливание вычислительного процесса.

Поскольку ЛЭА относится к алгоритмам динамического программирования (ДП), рассмотрим задание его вычислительной процедуры в виде множества взаимозависимых подзадач, результаты решения которых используются при решении больших подзадач, пока вся задача не будет решена. Зависимости между подзадачами в процедуре ЛЭА может быть представлена в виде **бесконтурного орграфа**, причем зачастую этот орграф задан неявно [14]. Вершины орграфа, лежащего

в основе алгоритмической схемы ЛЭА, соответствуют подзадачам, которые необходимо решить, а дуги орграфа — отношениям предшествования подзадач. Если имеются вершины $u_1, \ldots, u_k$, от которых стрелки указывают на v, то это означает: подзадача v может быть решена лишь после нахождения решений для подзадач $u_1, \ldots, u_k$ (рис. 1).

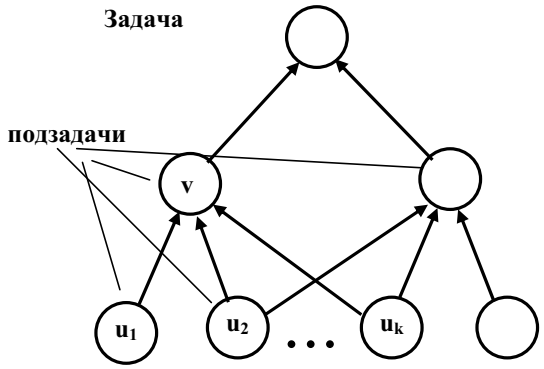

Рис. 1. Орграф подзадач вычислительной схемы ЛЭА.

Отметим, что в [14] анализируются бесконтурные орграфы для алгоритмов ДП решения различных дискретных задач.

2.2. Локальные алгоритмы элиминации переменных

Рассмотрим решение разреженной дискретной задачи оптимизации (1.1) – (1.4), структура которой описывается неориентированным графом взаимосвязей переменных $G = (X, E)$, с помощью локального алгоритма элиминации переменных. ЛА использует упорядочение вершин графа:

Определение 10. Упорядочением α вершин графа $G = (X, E)$ называется биекция (взаимно-однозначное отношение) $\alpha : X \leftrightarrow \{1, 2, \ldots, n\}$, где $n = |X|$.

В дальнейшем с целью упрощения обозначений будем использовать упорядочение α для индексирования множества вершин X графа G, т.е. $\alpha(x_i) = i$ и i будет рассматриваться как индекс или метка вершины x_i. G_α и X_α обозначают соответственно упорядоченный граф и упорядоченное множество вершин.
Для данного упорядочения α вершин графа G в виде $x_1, \ldots, x_n$ через $\mathcal{L}_i$ обозначим множество вершин с индексами из α, большими $i - 1$:

$$\mathcal{L}_i = \{x_i, x_{i+1}, \ldots, x_n\}$$

Рассмотрим неориентированный граф $G = (X, E)$, упорядочение α вершин X и вершину $y \in X$.

Определение 11. ([20]). **Монотонной окрестностью** вершины x_i называется множество вершин, соседних с x_i, с индексами, большими, чем i (согласно упорядочению α):

$$\overline{Nb}^{\alpha}_G(x_i) = \{x_j \in Nb_G(x_i) | j > i\} = Nb_G(x_i) \cap \mathcal{L}_{i+1}.$$

Определение 12. ([28], [29]). Граф, полученный из графа $G = (X, E)$ с помощью

- удаления вершины x и всех ребер, исходящих из нее, и
- соединения ребрами всех ранее не соседних вершин в $Nb(x)$,

называется x–**элиминационным** графом графа G и обозначается G^x. Описанная операция называется **элиминацией** вершины x.

Для данного упорядочения $x_1, x_2, \ldots, x_n$ вершин графа работа ЛЭА состоит в последовательной элиминации вершин $x_1, x_2, \ldots, x_n$ в текущем графе, изменении графа и вычислении соответствующей локальной информации о вершинах $h_i(Nb(x_i))$ [15]. Этот процесс порождает последовательность графов:

$$G^0 = G, G^1, \ldots, G^{j-1}, G^j, \ldots, G^n,$$

так что G^j — x_j–элиминационный граф графа G^{j-1}, а G^n — пустой граф.

Возникает вопрос, зависит ли вид элиминационного графа от порядка прохождения вершин. Ответ дан в следующей теореме:

Теорема 1. ([19]) *Любой граф, полученный из $G = (X, E)$ с помощью поочередной элиминации всех вершин из некоторого подмножества $Y \subset X$, не зависит от порядка элиминации вершин.*

Процесс преобразования графа взаимосвязей переменных, соответствующий процедуре элиминации переменных с помощью локального алгоритма, известен как **«элиминационная игра»**, которая впервые была введена Партером [28], как графовая интерпретация метода исключения Гаусса. Входом для алгоритма элиминационной игры является граф G и упорядочение α его вершин.
Алгоритм элиминационной игры состоит в следующем. Выбирается первая, согласно упорядочению α, вершина x, добавляются, если нужно, необходимые ребра, чтобы все вершины, соседние с x, образовывали клику. Затем удаляем вершину x из измененного графа и повторяем процесс, выбирая следующую вершину нового графа согласно упорядочению α. После прохождения всех вершин добавляем в исходном графе G все добавленные на каждом шаге ребра. Результатом этого алгоритма является пополненный граф $G^+_\alpha = (V,\ E^+_\alpha)$, в котором, по сравнению с исходным графом G, добавлены ребра.

В дальнейшем нам потребуется следующая

Лемма 1. (Parter [28]) $(x_i,\ x_j) \in E^+_\alpha \ \Leftrightarrow\ (x_i,\ x_j) \in E$ *или* $(x_i,\ x_k) \in E^+_\alpha$ *и* $(x_k,\ x_j) \in E^+_\alpha$ *для некоторого* $k < \min(i, j)$.

Введем важное для локального алгоритма элиминации понятие **элиминационного дерева**, введенное ранее в линейной алгебре [27] при разработке алгоритмов решения разреженных систем линейных уравнений и адаптированное для наших целей.

Определение 13. Элиминационным деревом (ЭД) графа $G = (X, E)$ для упорядочения вершин α называется ориентированное дерево $\overrightarrow{T}_\alpha$, 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 вершина (согласно упорядочению α) из монотонной окрестности $\overline{Nb}^{\alpha}_{G^+_\alpha}(x)$ вершины x в пополненном графе G^+_α.

Нетрудно видеть, что элиминационное дерево является орграфом вычислительной процедуры ЛЭА. Используя введенное в определении 13 отношение «предок–потомок», можно ввести понятие ориентированного пополненного графа $\overrightarrow{G}^+_\alpha$.

Свойства элиминационного дерева

Элиминационное дерево имеет то же самое множество вершин, что и граф G, поэтому является **стягивающим деревом** пополненного графа G^+_α. Вершина x_n принимается в качестве корня элиминационного дерева. Структура элиминационного дерева может быть задана вектором $PARENT[j] = \min\{i > j | (x_i, x_j) \in E^+_\alpha\}$, описывающим отношение «предок-потомок».

Из определения вектора $PARENT$ следует

Предложение 1. ([27]) *Если вершина x_i — непосредственный предок вершины x_j в элиминационном дереве, то $i > j$.*

Элиминационное дерево обладает следующими свойствами:

Предложение 2. ([27]) *Элиминационное дерево $\overrightarrow{T}_\alpha$ связного графа G является транзитивной редукцией ориентированного пополненного графа $\overrightarrow{G}^+_\alpha$.*

Теорема 2. ([27]) *Элиминационное дерево $\overrightarrow{T}_\alpha$ связного графа G является **деревом поиска в глубину** пополненного графа G^+_α.*

Доказательство. Рассмотрим упорядочение $x_1, \ldots, x_n$ вершин пополненного графа G^+_α. Рассмотрим поиск в глубину, начиная с x_n, использующий следующее правило: при наличии нескольких вариантов выбора вершин для перехода из данной, всегда выбирается вершина с **наибольшим** индексом. Поскольку дерево поиска в глубину и элиминационное дерево являются стягивающими деревьями пополненного графа, достаточно показать, что каждое ребро дерева поиска в глубину является также ребром элиминационного дерева. Действительно, рассмотрим ребро $(x_p,\ x_j)$ в дереве поиска в глубину; это значит, что во время поиска в глубину ребро $(x_p\ , x_j)$ в пополненном графе ведет в вершину x_j. С помощью математической индукции по расстоянию d вершины x_p от корня x_n и леммы 1 несложно

показать, что $p > j$. Понятно, что $x_p \in \overline{Nb}^{\alpha}_{G^+_\alpha}(x_j)$. Далее, методом доказательства от противного, можно показать, что $p = \min\{i \mid x_i \in \overline{Nb}^{\alpha}_{G^+_\alpha}(x_j)\}$.

3. Булева задача SAT и ее решение с помощью ЛЭА

Проиллюстрируем введенные графовые структуры на примере задачи SAT (satisfiability) [25] (задачи проверки выполнимости формулы логики высказываний), имеющей важное прикладное значение, причем приложения находятся в области автоматического вывода, тестирования электронных схем, баз данных, проектирования компьютеров, анализа изображений, САПР и робототехники [25]. Именно с задачи SAT Cook в 1971 г. начал исследование задач на NP–полноту [21].

Булева задача SAT — это формула, состоящая из трех компонентов:

- множества n переменных: $X_1, \ldots, X_n$.
- множества литералов (литерал — это переменная (X) или отрицание переменной ($\overline{X}$).
- множества m различных дизъюнктов: $C_1, \ldots, C_m$, каждый из которых состоит из литералов, соединенных конъюнкцией ($\vee$).

Цель задачи SAT состоит в установлении того, существуют ли логические значения {0,1} переменных, которые делают конъюнктивную нормальную форму (КНФ) $C_1 \wedge \ldots \wedge C_m$ истинной.

Задача SAT может рассматриваться как частный случай задачи удовлетворения ограничений, если рассматривать каждый дизъюнкт как ограничение. В [13] для решения этой задачи был использован оператор элиминации — **резолюция**, которая по двум дизъюнктам $(\alpha \vee Q)$ и $(\beta \vee \neg Q)$ выводит дизъюнкт $(\alpha \vee \beta)$, называемый **резольвентой**, в которой литерал Q элиминирован. Оператор элиминации (в данном случае — резолюция) порождает новые дизъюнкты, которым соответствуют новые ребра в графе взаимосвязей. Здесь мы будем использовать эквивалентную формулировку задачи SAT в виде задачи ДО без ограничений [25]:

$$\min_{x \in \{0,1\}^n} F(x) = \sum_{i=1}^{m} C_i(x),$$

где

$$C_i(x) = \prod_{j=1}^{n} Q_{ij}(x_j),\ Q_{ij}(x_j) = \begin{cases} 1 - x_j, & \text{если } X_j \text{ входит в } C_i; \\ x_j, & \text{если } \overline{X}_j \text{ входит в } C_i; \\ 1, & \text{в противном случае.} \end{cases}$$

Здесь псевдобулевы значения $x_j = 1$ и $x_j = 0$ соответствуют булевым значениям **истина** и **ложь**.

Рассмотрим пример решения задачи SAT, заданной в виде КНФ, состоящей из 4 дизъюнктов:

$$(X_1 \vee X_2 \vee \overline{X}_3) \wedge (\overline{X}_2 \vee X_3 \vee X_4) \wedge (\overline{X}_2 \vee X_5) \wedge (\overline{X}_3 \vee X_6 \vee X_7). \qquad (3.1)$$

Эквивалентная формулировка этой задачи SAT в виде задачи ДО без ограничений следующая:

$$F(x) = (1-x_1)\cdot(1-x_2)\cdot x_3 + x_2\cdot(1-x_3)\cdot(1-x_4) + x_2\cdot(1-x_5) + x_3\cdot(1-x_6)\cdot(1-x_7) \to \min,$$

где $x_j = 0,1$, $j = 1,\dots,7$, причем исходная задача SAT имеет решение при $\min_{x\in\{0,1\}^7} F(x) = 0$.

Используем ЛА элиминации переменных для решения этой задачи ДО без ограничений, используя порядок элиминации $\{x_5, x_2, x_1, x_4, x_3, x_6, x_7\}$. Структура формулы описывается графом взаимосвязей — неориентированным графом, вершины которого соответствуют переменным, причем ребро соединяет две вершины, если соответствующие переменные входят в один и тот же компонент формулы. Элиминационные графы, соответствующие элиминационному процессу, показаны на рис. 2. Вначале элиминируем первую (согласно упорядочению α) переменную x_5. $Nb(x_5) = \{x_2\}$. Решим следующую задачу ДО, используя полный перебор бинарных переменных из $Nb(x_5) = \{x_2\}$: $h_5(x_2) = \min_{x_5}\{x_2 \cdot (1 - x_5)|\ x_2 = 0,1\}$, и запомним оптимальное локальное решение x_5 как функцию $Nb(x_5)$, т.е. $x_5^*(x_2)$. В результате построим таблицу 1.

На следующем шаге элиминируем переменную x_2. Найдем окрестность $Nb(x_2)$, используя граф взаимосвязей, показанный на рис. 2 б). $Nb(x_2) = \{x_1, x_3, x_4\}$. Рассмотрим задачу ДО, соответствующую элиминации переменной x_2:

$$h_2(x_1, x_3, x_4) = \min_{x_2}\{h_5(x_2) + (1 - x_1)\cdot(1 - x_2)\cdot x_3 + x_2\cdot(1 - x_3)\cdot(1 - x_4)|$$

$$x_j = 0,1,\ j = 1,\dots,4\}.$$

Построим соответствующую таблицу 2.

Таблица 1.
Вычисление $h_5(x_2)$

x_2	h_5	x_5^*
0	0	0,1
1	0	0

Таблица 2.
Вычисление $h_2(x_1, x_3, x_4)$

x_1	x_3	x_4	h_2	x_2^*	x_1	x_3	x_4	h_2	x_2^*
0	0	0	0	0	1	0	0	0	0
0	0	1	0	0,1	1	0	1	0	0,1
0	1	0	0	1	1	1	0	0	0,1
0	1	1	0	1	1	1	1	0	0,1

Элиминируем x_1. $Nb(x_1) = \{x_3, x_4\}$ (рис. 3 в)). Решим подзадачу:

$$h_1(x_3, x_4) = \min_{x_1}\{h_2(x_1, x_3, x_4)|\ x_j = 0,1,\ j = 1,3,4\}.$$

Построим таблицу 3.

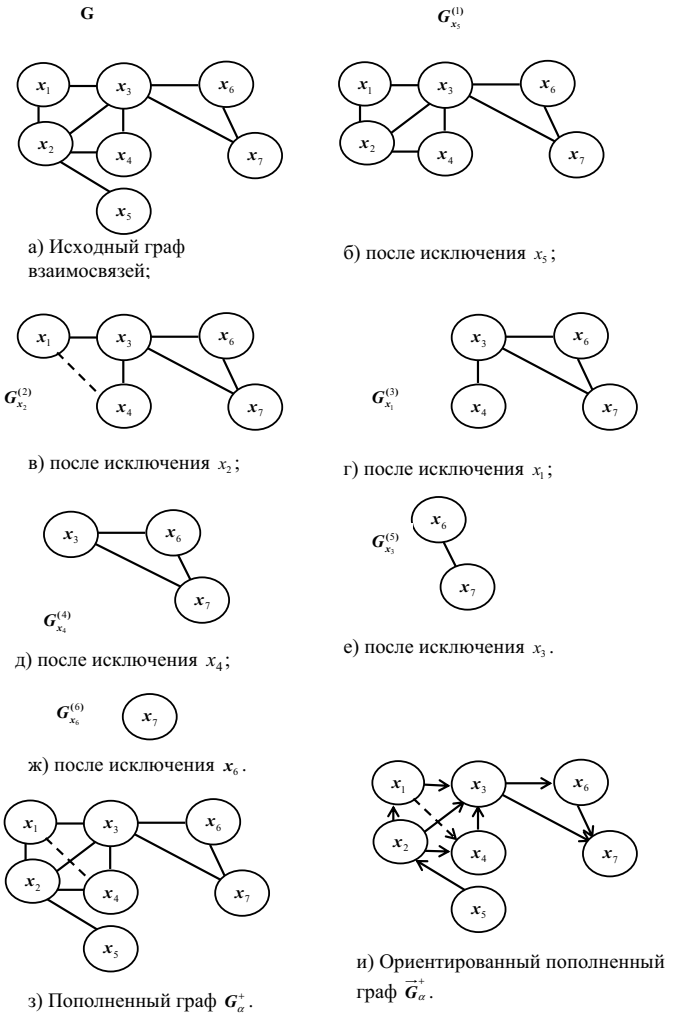

Рис. 2. Элиминационная игра. Пополнение показано пунктиром.

Элиминируем переменную x_4. $Nb(x_4) = \{x_3\}$ (рис. 3 г)). Решим подзадачу:

$$h_4(x_3) = \min_{x_4}\{h_1(x_3, x_4)|\ x_j = 0, 1,\ j = 3, 4\}.$$

Построим таблицу 4. Элиминируем переменную x_3. $Nb(x_3) = \{x_6, x_7\}$ (рис. 3 д)). Решим подзадачу:

$$h_3(x_6, x_7) = \min_{x_3}\{h_4(x_3) + x_3 \cdot (1 - x_6) \cdot (1 - x_7)|\ x_j = 0, 1,\ j = 3, 6, 7\}.$$

Построим таблицу 5.

Таблица 3.
Вычисление $h_3(x_6, x_7)$

x_3	x_4	h_1	x_1^*	x_3	x_4	h_1	x_1^*
0	0	0	0,1	1	0	0	0,1
0	1	0	0,1	1	1	0	0,1

Таблица 4.
Вычисление $h_4(x_3)$

x_3	h_4	x_4^*
0	0	(0,1)
1	0	(0,1)

Таблица 5.
Вычисление $h_3(x_6, x_7)$

x_6	x_7	h_3	x_3^*	x_6	x_7	h_3	x_3^*
0	0	0	0	1	0	0	0,1
0	1	0	0,1	1	1	0	0,1

Таблица 6.
Вычисление $h_6(x_7)$

x_7	h_6	x_6^*
0	0	(0,1)
1	0	(0,1)

Элиминируем x_6. $Nb(x_6) = \{x_7\}$ (рис. 3 е)). Решим задачу
$h_6(x_7) = \min_{x_6}\{h_3(x_6, x_7)|\ x_j = 0, 1,\ j = 6, 7\}$.
Построим таблицу 6. Элиминируя x_7, получим $h_7 = \min_{x_7}\{h_6(x_7)\} = 0$ для любого бинарного x_7.

Итак, КНФ (3.1) истинна для некоторых значений переменных. Для нахождения решения необходимо выполнить обратную часть процедуры ЛЭА. Последовательно найдем $x_7, x_6, x_3, x_4, x_1, x_2, x_5$, т.е. оптимальные частичные (локальные) решения из сохраненных таблиц 6, 5, 4, 3, 2, 1. Так как решений очень много, перечислим лишь некоторые из них:

Таблица 7. Некоторые решения задачи SAT

x_5	x_2	x_1	x_4	x_3	x_6	x_7
0,1	0	0,1	0	0	0,1	0,1
0,1	0	0	1	0	0,1	0,1
1	1	0	1	1	0,1	1
1	1	0	1	1	1	0,1
1	1	0	1	0	0,1	0,1
1	1	1	0,1	1	0,1	1
0,1	0	1	0,1	1	0,1	1

Элиминационное дерево для рассматриваемой задачи приведено на рис.3 а).
Следует отметить, что ЛЭА является алгоритмом не только элиминации переменных, но и **вычисления информации**. На рис. 3 б) показано, какая именно информация вычисляется при элиминации переменных.

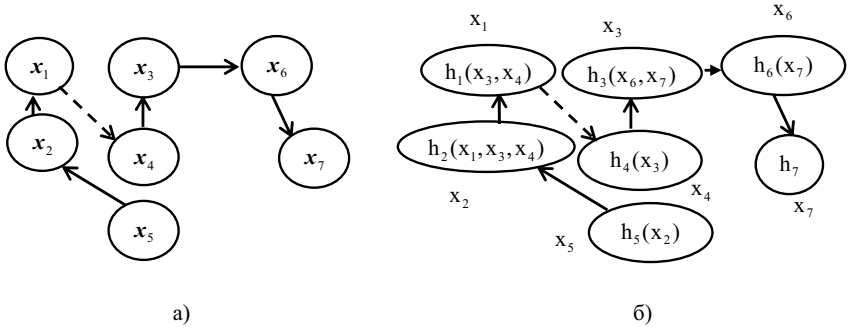

Рис. 3. а) Элиминационное дерево для задачи SAT;
б) Вычисление информации, соответствующее элиминации переменных.

4. Использование множеств достижимости для описания процесса вычисления информации с помощью ЛЭА

Выше процесс вычисления информации с помощью ЛЭА описывался в виде последовательности элиминационных графов

$$G^0 = G, G^1, \ldots, G^n,$$

где $G^0 = G,\ G^i = (G^{i-1})_{x_i} = (X_i, E_i),\ i = 2, \ldots, n$. Здесь $X_i = \{x_{i+1}, \ldots, x_n\}$. (см. рис. 2). Граф G^i отражает структуру дискретной задачи, оставшейся для решения после i-го шага ЛЭА.

ЛЭА вычисляет информацию, обрабатывая окрестности элементов в изменяющихся элиминационных графах (см. рис. 2). При изучении ЛЭА может быть полезно использование так называемых **множеств достижимости** (reachable sets) [24], являющихся обобщением обычной окрестности, что позволяет производить локальные вычисления на основе анализа исходного графа, а не элиминационных графов.

Определение 14. Пусть S — подмножество множества X вершин графа $G = (X, E)$ и $x \notin S$. Говорят, что вершина x **достижима** из вершины y **через** S, если существует путь $y, s_1, \ldots, s_k, x$ из y в x такой, что $s_i \in S$ для $i = 1, \ldots, k$.

Замечание 1. В этом определении возможен случай $k = 0$, так что любая вершина, смежная y и не принадлежащая S, достижима из y через S.

Пример 1. Для графа G (рис. 2 а)) и $S = \{x_3,\ x_6\}$ вершины $x_2,\ x_4,\ x_7$ достижимы из x_1 через S.

Определение 15. Множество достижимости $Reach(y, S)$ вершины y через подмножество S определяется следующим образом:

$$Reach(y, S) = \{x \in X \setminus S \mid x \text{ достижима из вершины } y \text{ через } S\}.$$

Пример 2. Для графа G (рис. 2 а)) и $S = \{x_3,\ x_6\} : Reach(x_2,\ S) = \{x_1,\ x_4,\ x_5,\ x_7\}$.

Замечание 2. Понятие достижимости **симметрично**, т. е. для различных $x, y \in X \setminus S$ справедливо $x \in Reach(y, S) \Longleftrightarrow y \in Reach(x, S)$.

Замечание 3. Понятие множества достижимости обобщает понятие окрестности, так как $Nb(y) = Reach(y, \emptyset)$.

Оказывается, что окрестность любой вершины элиминационного графа может быть легко найдена через некоторое множество достижимости исходного графа, что видно из следующей теоремы:

Теорема 3. ([24]) *Для любой вершины y элиминационного графа $G^i = (X_i, E_i)$ окрестность $Nb_{G^i}(y)$ вершины y равна*

$$Nb_{G^i}(y) = Reach(y, \{x_1, \ldots, x_i\}),$$

где оператор достижимости Reach применяется к исходному графу $G^0 = G$.

Другое важное свойство множеств достижимости позволяет найти множество ребер E_α^+ пополненного графа $G_\alpha^+ = (X, E_\alpha^+)$:

Теорема 4. ([24]) *Для пополненного графа* $G_\alpha^+ = (X, E_\alpha^+)$*:*

$$E_\alpha^+ = \{(x_i, x_j) |\ x_j \in Reach(x_i, \{x_1, \ldots, x_{i-1}\})\},$$

где оператор достижимости Reach применяется к исходному графу $G^0 = G$*.*

Замечание 4. Эффективность работы ЛЭА существенно зависит от порядка, в котором элиминируются переменные. К сожалению, задача поиска наилучшего упорядочения (в смысле минимизации числа пополненных ребер) является NP–полной [31]. В связи с этим при поиске достаточно хорошего упорядочения вершин графа взаимосвязей целесообразно использовать эвристические алгоритмы, такие как "Minimum Degree" [16].

Замечание 5. Используя элиминационное дерево и бесконтурный орграф вычислительной процедуры ЛЭА (см. рис. 3 б)), можно построить древовидную декомпозицию [12], [15] задачи ДО.

Заключение

Локальный элиминационный алгоритм вычисления информации — перспективный подход, делающий возможным решение прикладных дискретных разреженных задач оптимизации. Использование графовых структур позволяет рационально построить вычислительную схему локального алгоритма. *Перспективными направлениями* дальнейших исследований являются разработка эффективных схем локального элиминационного алгоритма и использование соответствующих графовых структур при решении конкретных задач дискретной оптимизации, обладающих специальной структурой.

Список цитируемых источников

1. *Евстигнеев В.А.* Применение теории графов в программировании. — М.: Наука, 1985. — 352 с.
2. *Евстигнеев В.А., Касьянов В.Н.* Толковый словарь по теории графов в информатике и программировании. — Новосибирск: Наука, 1999. — 288 с. (электронная версия словаря — http://pco.iis.nsk.su/grapp).
3. *Емеличев В.А., Мельников О.И., Сарванов В.И., Тышкевич Р.И.* Лекции по теории графов. — М.: Наука, 1990. — 384 с.
4. *Журавлев Ю.И.* Избранные научные труды. — М.: Магистр, 1998. — 420 с.
5. *Журавлев Ю.И., Лосев Г.Ф.* Окрестности в задачах дискретной математики // Кибернетика и системный анализ. — 1995. — №2. — С. 32–41.
6. *Леонтьев В.К.* Дискретная оптимизация // Журнал вычислительной математики и математической физики. — 2007. — 47. — С. 338–352.
7. *Свами М., Тхуласираман К.* Графы, сети и алгоритмы. — М.: Мир, 1984. — 455 с.

8. *Сергиенко И.В., Шило В.П.* Задачи дискретной оптимизации: проблемы, методы решения, исследования. — К.: Наукова думка, 2003. — 162 с.

9. *Щербина О.А.* О локальных алгоритмах решения задач дискретной оптимизации // Проблемы кибернетики. — 1983. — N 40. — P. 171–200.

10. *Щербина О.А.* О несериальной модификации локального алгоритма декомпозиции задач дискретной оптимизации // Динамические системы. — 2005. — Вып.19. — С. 179–190.

11. *Щербина О.А.* Элиминационные алгоритмы декомпозиции задач дискретной оптимизации // Таврический вестник информатики и математики. — 2006. — №2. — С. 28–41.

12. *Щербина О.А.* Локальные алгоритмы и древовидная декомпозиция для задач дискретной оптимизации // Динамические системы. — 2006. — Вып. 20. — С. 89–103.

13. *Щербина О.А.* Локальные элиминационные алгоритмы для задач удовлетворения ограничений // Таврический вестник информатики и математики. — 2007. — №1. — С. 24–39.

14. *Щербина О.А.* Методологические аспекты динамического программирования // Динамические системы. — 2007. — Вып. 22. — С. 21–36.

15. *Щербина О.А.* Локальные элиминационные алгоритмы решения разреженных дискретных задач // Журнал вычислительной математики и математической физики. — 2008. — том 48, N 1. — С. 161—177.

16. *Amestoy P.R., Davis T.A., Duff I.S.* An approximate minimum degree ordering algorithm // SIAM Journal on Matrix Analysis and Applications. — 1996. — V. 17, N.4. — P. 886–905.

17. *Arnborg S., Corneil D.G., Proskurowski A.* Complexity of finding embeddings in a k-tree // SIAM J. Alg. Disc. Meth. — 1987. — V.8. — P. 277–284.

18. *Beeri C., Fagin R., Maier D., Yannakakis M.* On the desirability of acyclic database schemes // J. ACM. — 1983. — V. 30. — P. 479–513.

19. *Bertele U., Brioschi F.* Nonserial Dynamic Programming. — New York: Academic Press, 1972. — 235 p.

20. *Blair J.R.S., Peyton B.W.* An introduction to chordal graphs and clique trees // Graph theory and sparse matrix computation / A. George et al. (eds.). — New York: Springer, 1993. — P. 1–29.

21. *Cook S.A.* The complexity of theorem-proving procedures // Proc. 3rd Ann. ACM Symp. on Theory of Computing Machinery. — New York, 1971. — P. 151–158.

22. *Courcelle B.* Graph rewriting: An algebraic and logic approach // Handbook of Theoretical Computer Science, Volume B /J. Van Leeuwen (ed.). — Amsterdam et al.: Elsevier, 1990. — P. 193–242.

23. *Dechter R.* Bucket elimination: A unifying framework for reasoning // Artificial Intelligence. — 1999. — V. 113. — P. 41–85.

24. *George A., Liu J.W.H.* Computer Solution of Large Sparse Positive Definite Systems. — Englewood Cliffs: Prentice-Hall Inc., 1981. — 324 p.

25. *Gu J., Purdom P.W., Franco J., Wah B.W.* Algorithms for the satisfiability (SAT) problem: A survey // Satisfiability Problem Theory and Applications /D. Du, J. Gu, and P.M. Pardalos (eds.). — Providence, RI: Amer. Math. Soc., 1997. — P. 19–153.

26. *Hicks I.V., Koster A.M.C.A., Kolotoglu E.* Branch and tree decomposition techniques for discrete optimization // Tutorials in Operations Research. — New Orleans: INFORMS, 2005. — P. 1—29. — http://ie.tamu.edu/People/faculty/Hicks/bwtw .pdf
27. *Liu J.W.H.* The role of elimination trees in sparse factorization // SIAM Journal on Matrix Analysis and Applications. — 1990. — V. 11. — P. 134–172.
28. *Parter S.* The use of linear graphs in Gauss elimination // SIAM Review. — 1961. — V. 3. — P. 119–130.
29. *Rose D.J.* A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations // Graph Theory and Computing /R.C. Read (ed.). — New York: Academic Press, 1972. — P. 183–217.
30. *Tarjan R.E.* Depth first search and linear graph algorithms // SIAM J. Comput. — 1972. — V. 1. — P. 146–160.
31. *Yannakakis M.* Computing the minimum fill-in is NP-complete // SIAM J. Alg. Disc. Meth. — 1981. — V. 2. — P. 77-–79.

Получена 02.06.2008