Динамические системы, вып. 20 (2006), 89–103

ДИНАМИЧЕСКИЕ СИСТЕМЫ
Межведомственный научный сборник

УДК 519.68

Локальные алгоритмы и древовидная декомпозиция для задач дискретной оптимизации [1]

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

Аннотация. Рассмотрен класс разреженных задач дискретной оптимизации и обсуждаются возможности выделения блочно-древовидной структуры. Рассмотрено применение локального алгоритма декомпозиции для решения этих задач дискретной оптимизации.

1. Введение

Задачи дискретной оптимизации (ДО) естественным образом возникают при создании систем автоматизации проектирования, автоматизированных систем планирования ресурсов (ERP), при решении задач логистики, в частности, при оптимизации цепочек предложения (Supply Chain Management), решении задач искусственного интеллекта, робототехники и т. п. Это обусловлено тем, что дискретные оптимизационные модели адекватно отражают нелинейные зависимости, неделимость объектов, учитывают ограничения логического типа и всевозможные технологические, в том числе и имеющие качественный характер, требования. К сожалению, большинство интересных для практических приложений задач ДО являются NP-трудными и, кроме того, содержат большое число переменных и/или ограничений, что создает сложности при попытке решения этих задач с помощью современных решателей.

Однако обычно прикладные задачи ДО имеют специальную структуру, и их матрицы ограничений являются разреженными. Перспективными декомпозиционными методами, использующими разреженность матрицы ограничений задач ДО, представляются элиминационные алгоритмы декомпозиции, включающие локальные алгоритмы декомпозиции [4], [6], [5], алгоритмы несериального динамического программирования (НСДП) ([12], [29], [31]), алгоритмы сегментной элиминации [18]. Для выделения специальных структур задач ДО представляются перспективными такие графовые декомпозиционные подходы как методы древовидной

[1]Работа выполнена при финансовой поддержке австрийского фонда FWF, грант No. P19138-N13

декомпозиции (ДД) [28]. Несмотря на обширную библиографию по этой тематике за рубежом, в отечественной литературе публикаций по ДД практически нет (можно отметить лишь близкие по тематике работы по локальным алгоритмам декомпозиции [5], [6]). Важность и актуальность указанных подходов в ДО обусловлены результатами Courcelle [17], Arnborg et al. [8], 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 ДО, имеющих ограниченную древовидную ширину. Методы ДД показали свою эффективность при решении задач ДО: задач кольцевой маршрутизации [15], задачи коммивояжера [16], задачи оптимального назначения радиочастот [30].

Настоящая статья описывает возможности совместного использования методов древовидной декомпозиции и локальных алгоритмов декомпозиции при решении задач дискретной оптимизации.

2. Задачи дискретной оптимизации и их графовые представления

Рассмотрим задачу ДО с ограничениями

$$F(x_1, x_2, \ldots, x_n) = \sum_{k \in K} f_k(Y^k) \to \max \tag{2.1}$$

при ограничениях

$$g_i(X^i)\ R_i\ 0, \quad i \in M = \{1, 2, \ldots, m\}, \tag{2.2}$$

$$x_j \in D_j, \quad j \in \{1, \ldots, n\}, \tag{2.3}$$

где

$$Y^k \subseteq \{x_1, x_2, \ldots, x_n\}, \quad k \in K = \{1, 2, \ldots, t\}\,; \tag{2.4}$$

$$X^i \subseteq \{x_1, x_2, \ldots, x_n\}, \quad R_i \in \{\leq, =, \geq\}, \quad i \in M = \{1, 2, \ldots, m\}. \tag{2.5}$$

Рассмотрим пример задачи ДО с ограничениями:

Пример 1.

$$\begin{aligned}
&2x_1 + 3x_2 + x_3 + 5x_4 + 4x_5 + 6x_6 + x_7 && \to \max\\
&3x_1 + 4x_2 + x_3 && \leq 6,\ (C_1)\\
&\quad 2x_2 + 3x_3 + 3x_4 && \leq 5,\ (C_2)\\
&\quad 2x_2 \qquad + 3x_5 && \leq 4,\ (C_3)\\
&\qquad 2x_3 \qquad + 3x_6 + 2x_7 && \leq 5,\ (C_4)\\
&x_j = 0, 1,\ j = 1, \ldots, 7.
\end{aligned}$$

Определение 1 ([12], [5]). Две переменные $x \in X$ и $y \in Y$ взаимосвязаны в задаче ДО с ограничениями, если они появляются вместе в одной компоненте ЦФ или в одном и том же ограничении (другими словами, если переменные входят одновременно во множество X^i или во множество Y^k).

Введем графовое представление задачи ДО. Наиболее адекватным представлением структуры задачи ДО является гиперграфовое представление, в котором множество вершин гиперграфа H соответствует множеству переменных X задачи ДО, а гиперребра гиперграфа образуют подмножества взаимосвязанных переменных.

Часто вместо гиперграфового представления используется более наглядное представление структуры задачи ДО с помощью *двойственного* и *первичного* графов. Первичный граф называется обычно графом взаимосвязей (interaction graph [12]) задачи ДО.

Определение 2. *Двойственным графом* гиперграфа $H = (V, S)$ называется граф, вершины которого соответствуют гиперребрам гиперграфа, причем пара таких вершин соединяется ребром в двойственном графе, если они имеют общие вершины из V (см. рис. 1).

Определение 3. *Первичным графом* гиперграфа $H = (V, S)$ является граф, у которого множество вершин совпадает с V и две вершины соединены ребром, если они входят в одно и то же гиперребро (см. рис. 2).

Нам потребуется понятие графа взаимосвязей, естественным образом представляющего структуру задачи ДО, который является первичным графом гиперграфа $H = (V, S)$ задачи ДО.

Определение 4. *Графом взаимосвязей* (interaction graph [12]) задачи ДО называется неориентированный граф $G = (V, E)$, для которого

1. множество вершин V соответствует множеству переменных X задачи оптимизации;

2. две вершины графа соединены ребром тогда и только тогда, когда соответствующие им переменные взаимосвязаны.

В дальнейшем будем отождествлять переменные задачи ДО с вершинами графа взаимосвязей.

Определение 5. Множество переменных, взаимосвязанных с переменной $x \in X$ в графе взаимосвязей задачи ДО, называется *окрестностью* переменной и обозначается $Nb(x)$ или $Nb_G(x)$.

В дальнейшем нам потребуются следующие понятия.

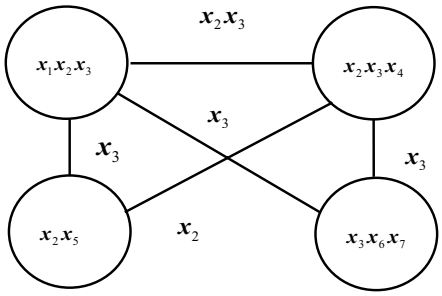


Рис. 1. Двойственный граф для задачи ДО из примера 1.

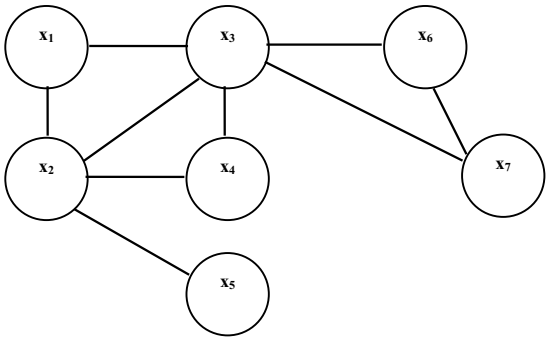


Рис. 2. Первичный граф (взаимосвязей) в задаче ДО из примера 1.

Определение 6 ([2]). *Клика* (Clique) – подмножество $V' \subseteq V$ вершин графа $G = (V, E)$, в котором любые две вершины смежны, т.е. порожденный им подграф $G(V')$ является полным.

Определение 7 ([2]). Клика C в графе G называется *максимальной*, если C не является подмножеством никакой другой клики в G .

Определение 8 ([1]). *Стягивающее дерево* [[2]]– для данного неориентированного графа дерево, имеющее то же множество вершин, что и сам граф.

Определение 9 ([2]). *Граф клик* - это граф пересечений максимальных клик графа, причем вершины его соответствуют максимальным кликам и две вершины смежны тогда и только тогда, когда соответствующие клики имеют непустое пересечение.

Пусть $K_G = \{C_i\}$ - множество максимальных клик для графа G и V_i – множество вершин для максимальной клики C_i.

Определение 10. *Деревом клик* для графа G называется дерево $T = (K_G, \mathcal{E})$, в котором множество ребер $\mathcal{E}$ удовлетворяет следующему свойству: для любых двух клик $C_i = (V_i, E_i)$, $C_j = (V_j, E_j) \in K_G$ и любой клики $C_k = (V_k, E_k) \in K_G$, находящейся на пути из C_i к C_j на T, выполняется включение $V_i \bigcap V_j \subseteq V_k$.

Определение 11. Вершина графа называется *симплициальной*, если она со своими смежными вершинами образует клику.

3. Древовидная декомпозиция задач дискретной оптимизации. Хордальные графы, триангуляция

Одним из известных классов графов являются хордальные графы (ХГ) [25], которые в некотором смысле являются обобщениями деревьев [22]. Так, ХГ имеют структуру, подобную деревьям, и для них также имеются быстрые алгоритмы для поиска, элиминации и декомпозиции, они, как и деревья, широко используются в различных приложениях, таких как СУБД, проектирование схем и др. Исторически первой областью приложения ХГ были вычисления с разреженными матрицами [23].

Определение 12 ([25]). Граф называется *хордальным*, если он не содержит ни одной порожденной клики $C = (V_0, E_0)$, $|V_0| \geq 4$.

Другими словами, в ХГ каждый цикл длиной ≥ 4 обладает хордой (т.е. ребром, соединяющем 2 несмежные вершины цикла). Хордальные графы называются по-другому *триангулированными* графами или треугольными графами.

Определение 13. *Триангуляцией* графа $G = (V, E)$ называется ХГ $H = (V, F)$, содержащий G в качестве подграфа: $E \subseteq F$.

Процесс преобразования первичного графа взаимосвязей, соответствующий процедуре НСДП, известен как *«элиминационная игра»*, которая впервые была введена Партером [32], как графовая интерпретация метода исключения Гаусса. Входом для алгоритма элиминационной игры является граф G и упорядочение α вершин графа G (т.е. $\alpha(v) = i$, если v – i-я вершина в упорядочении α). Алгоритм состоит в следующем. Выбирается первая, согласно упорядочению α, вершина v, добавляем, если нужно необходимые ребра, чтобы все вершины, смежные с v, образовывали клику. Затем удаляем вершину v из измененного графа и продолжаем процесс, выбирая следующую вершину нового графа согласно упорядочению α. После прохождения всех вершин добавляем в исходном графе все добавленные на каждом шаге ребра. Результатом этого алгоритма является пополненный граф G^+_α, в котором, по сравнению с исходным графом, добавлены ребра.

Fulkerson & Gross [21] показали, что:

1. Пополненный граф G_α^+, порожденный с помощью элиминационной игры, является хордальным графом, т.е. триангуляцией графа G;

2. Пополненные графы, полученные с помощью элиминационной игры, образуют в точности класс хордальных графов.

Эти результаты показывают возможность построения древовидных декомпозиций для задачи ДО, используя элиминационную игру, так как для хордальных графов достаточно просто находятся максимальные клики и строится *дерево клик*, являющееся древовидной декомпозицией.

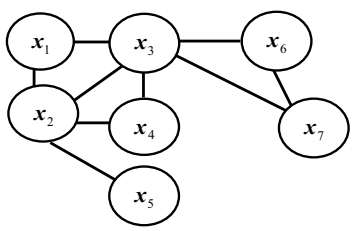
При использовании алгоритма элиминационной игры для триангуляции задач ДО с ограничениями следует отметить необходимость одновременного рассмотрения двойственного графа и приписывания ограничений соответствующим кликам (блокам).

Пополненные графы - результат обработки вершин графа G с помощью алгоритма элиминационной игры, существенно зависят от порядка элиминации α вершин. Для нахождения триангуляций с небольшим пополнением, перед использованием элиминационной игры необходимо найти хорошее упорядочение вершин данного графа. Отметим, что задача нахождения упорядочения вершин графа, обеспечивающего минимальное пополнение, является NP–трудной [36].

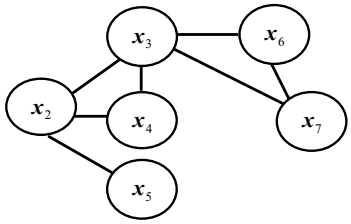
Интересные прикладные задачи на графах связаны с нахождением триангуляции графа, т.е. его вложения в ХГ. В общем случае для данного графа существует много триангуляций. При нахождении наилучшей триангуляции обычно ищут минимум различных графовых параметров триангуляции. Например, задача поиска триангуляции с минимальным пополнением ищет триангуляцию с наименьшим числом добавленных ребер.

Элиминационная игра может быть использована таким образом, чтобы упорядочение α порождалось в процессе работы алгоритма. В этом случае мы можем на каждом шаге i выбирать вершину v графа G^{i-1} согласно любому желательному для нас критерию, и положить $\alpha(v) = i$, определяя тем самым порядок элиминации α. Одной из хорошо известных эвристик является эвристика «Minimum Degree», которая выбирает вершину v с минимальной степенью в G^{i-1} на каждом шаге i.

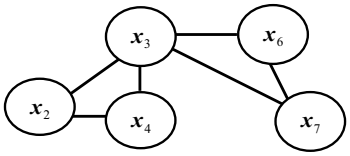
Одной из причин, по которой многие задачи оптимизации, которые являются трудными на общих графах, легко решаются на деревьях, является отсутствие циклов в деревьях. При решении задачи на дереве обычно возможно найти «локальные» или частичные решения для поддерева. Эти локальные решения можно использовать при решении задачи ДО с помощью локальных алгоритмов декомпозиции. Локальный алгоритм начинает с решения подзадач, соответствующих листьям дерева и переходит от меньших к большим подзадачам. Все это обусловливает перспективность поиска и выявления древовидных структур на графа, что может быть осуществлено и с помощью методов *древовидной декомпозиции* [33]. Понятия древовидной декомпозиции и древовидной ширины, являющейся мерой «древовидности» графа, были введены ROBERTSON, SEYMOUR [33].



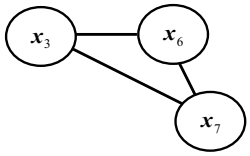
а) исходный граф;



б) после исключения переменной x_1.



в) после исключения переменной x_5.



г) после исключения переменных x_2 и x_4.

д) после исключения переменных x_6 и x_7.

Рис. 3. Элиминационная игра.

Алгоритмы древовидной декомпозиции ([14], [19], [24], [26]) используют реструктуризацию задачи, основанную на выделении древовидной структуры в графе взаимосвязей переменных задачи. При этом вершины древовидной структуры включают подмножества переменных исходной задачи ДО, каждое из которых рассматривается как единая вершина древовидной структуры.

Определение 14 ([14]). Для заданного графа $G = (V, E)$ *древовидной декомпозицией* (ДД) (tree decomposition) называется пара $(\{X_i \mid i \in I\},\ T = (I, F))$, где $\{X_i \mid i \in I\}$ - семейство подмножеств вершин v и T - дерево с множеством вершин i и множеством ребер $F \subseteq I \times I$ такими, что

1. $\bigcup_{i \in I} X_i = V$;
2. для всех ребер $(v, w) \in E$ существует такое $i \in I$, что $v \in X_i,\ w \in X_i$;
3. для всех $i, j, k \in I$ таких, что j лежит на пути в T из i в k, справедливо включение $X_i \bigcap X_k \subseteq X_j$.

Шириной ДД называется $\max_{i \in I} |X_i| - 1$. *Древовидная ширина* (ДШ) графа G определяется как наименьшая ширина древовидной декомпозиции G и обозначается $tw(G)$. В частности, если в этом определении рассматривать не деревья, а пути (точнее, цепи), то получим определение *путевой ширины* $pw(G)$. Так как

всякий путь есть дерево, то имеет место неравенство $tw(G) \leq pw(G)$. Графы с ДШ k известны также, как частичные k–деревья.

Задача поиска и наилучшей ДД состоит в нахождении триангуляции с минимальным размером наибольшей клики. Известно, что многие NP–полные задачи разрешимы за полиномиальное время, если они рассматриваются на графах с ограниченной ДШ [14], [33]. К сожалению, как задача поиска триангуляции с минимальным пополнением, так и задача поиска ДШ являются NP-трудными [8], [36]. Однако для обеих задач имеются полиномиально вычислимые альтернативы нахождения так называемой минимальной триангуляции [27].

4. Свойства хордальных графов

Лемма 1 ([20]). *Хордальный неполный граф содержит, по крайней мере, две несмежные симплициальные вершины.*

В [21] предложена элиминационная схема для распознавания ХГ на основе леммы 1: граф является хордальным тогда и только тогда, когда возможно повторно находить симплициальную вершину и удалять ее, пока не останется ни одной вершины. Время счета для такого тривиального алгоритма составляет $O(n^2m)$.

Теорема 1 ([21]). *ХГ с n вершинами имеет самое большее $n = |V|$ максимальных клик.*

Графы клик для графов общего вида могут содержать большое число клик (до C_n^k различных клик размера k). Поэтому ясно, что графы клик для графов общего вида может быть экспоненциально большими, а нахождение максимальных клик – NP–трудно. Свойства именно хордальных графов делают их графы клик чрезвычайно полезными. Для ХГ число максимальных клик не превосходит $|V|$, поэтому они могут быть найдены за линейное время [13]. После нахождения графа клик для ХГ возможно выделить из него дерево клик, являющееся ДД, с помощью следующего свойства:

Теорема 2 ([10], [34]). *Стягивающее дерево графа клик является деревом клик для ХГ G тогда и только тогда, когда оно является максимальным стягивающим деревом графа клик.*

Процедура решения задачи ДО с помощью ДД состоит из двух этапов:
1) нахождение достаточно хорошей ДД;
2) применение локального алгоритма декомпозиции для решения задачи.

5. Локальный алгоритм декомпозиции задач дискретной оптимизации с древовидной структурой

При изучении сложных объектов не всегда возможно получить (или вычислить) полную информацию об объекте в целом, поэтому представляет интерес получение информации об объекте, рассматривая его по частям, т.е. локально.

В [3] Ю.И. Журавлев описывает локальные алгоритмы вычисления информации. Локальный алгоритм (ЛА) изучает элементы в порядке, задаваемом алгоритмом упорядочения A_π, при этом используя (локальную) информацию об элементах из окрестности данного элемента.

ЛА декомпозиции задач ДО имеют свою специфику, состоящую в том, что они не вычисляют предикаты, а, используя принцип оптимальности Беллмана, вычисляют оптимальные частичные решения подзадач, соответствующих блокам задачи ДО. При этом они запоминают информацию не о предикатах, а о значениях переменных; такую информацию Ю.И.Журавлев предложил называть *индикаторной* информацией.

Рассмотрим ЛА [6] для решения задач ДО с древовидной структурой, т.е. для которых известна ДД, которую мы считаем заданной в виде множества блоков переменных и ограничений. ЛА решает задачу ДО, двигаясь снизу вверх, от листьев ДД к корню дерева T. Пусть $B_1 = \left(\bar{S}_1,\ U_1\right),\ \ B_2 = \left(\bar{S}_2,\ U_2\right), \ldots,\ \ B_k = \left(\bar{S}_k,\ U_k\right)$ - блоки ДД, где $\bar{S}_r,\ U_r$ являются соответственно множествами индексов переменных и ограничений для r-го блока, $r = 1, \ldots, k$, причем

$$\bigcup_{r=1}^{k} U_r = M = \{1, \ldots, m\},\ \ \bigcup_{r=1}^{k} S_r = N = \{1, \ldots, n\}, U_{r_1} \cap U_{r_2} = \emptyset,\ r_1 \neq r_2.$$

Рассмотрим вершину r дерева T и введем дерево D_r, состоящее из вершины r и всех ее потомков.

Введем необходимые обозначения:

$\bar{S}_r$ – множество индексов переменных, принадлежащих блоку B_r;

$S_{rr'}$ – множество индексов переменных, принадлежащих одновременно блокам B_r и $B_{r'}$, т.е. $S_{rr'} = \bar{S}_r \bigcap \bar{S}_{r'}$;

p_r – вершина–предок вершины r;

J_r – множество потомков вершины r.

Тогда $X_{S_{p_r r}}$ – вектор переменных, общих для блоков B_{p_r} и B_r (здесь $S_{p_r r} = \bar{S}_{p_r} \bigcap \bar{S}_r$).

Обозначим через Z_{D_r} следующую задачу: для каждого вектора $X_{S_{p_r r}}$ найти X_{S_r} и $X_{S_{rr'}}$, такие, что

$$h_{B_r}(X_{S_{p r r}}) = f_{D_r}\left(X_{S_{p r r}}\right) = \max_{X_{S_r}, X_{S_{rr'}}} \left\{ C_{S_r} X_{S_r} + \sum_{r' \in J_r} \left[f_{D_{r'}}(X_{S_{rr'}}) + C_{S_{rr'}} X_{S_{rr'}} \right] \right\} =$$

$$= \max_{X_{S_r}, X_{S_{rr'}}} \left\{ C_{S_r} X_{S_r} + \sum_{r' \in J_r} \left[h_{B_{r'}}(X_{\bar{S}_r \bigcap \bar{S}_{r'}}) + C_{S_{rr'}} X_{S_{rr'}} \right] \right\}$$

при ограничениях

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

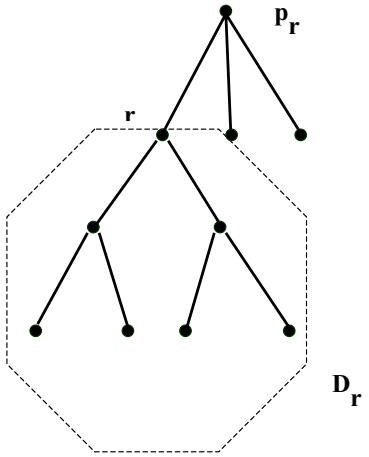


Рис. 4. Дерево D_r потомков

Здесь $f_{D_r}(X_{S_{p_rr}})$ – значение ЦФ задачи, соответствующей дереву D_r, $f_{D_{r'}}\left(X_{S_{rr'}}\right) = C_{D_{r'}}X_{D_{r'}}$. Можно это значение приписать корню $B_{r'}$ дерева D_r, т.е. записать: $h_{B_{r'}}(X_{\bar{S}_r \bigcap \bar{S}_{r'}}) = f_{D_{r'}}\left(X_{S_{rr'}}\right)$.

Работа ЛА $\mathfrak{A}_{BT}$ для решения задач ДО с древовидной структурой основана именно на том, что при фиксации вектора $X_{S_{p_rr}}$ задача распадается на две задачи: первая соответствует дереву D_r; а вторая — дереву $T - D_r$.

6. Решение конкретной задачи ДО с помощью выделения древовидной структуры и применения локального алгоритма декомпозиции

Рассмотрим для задачи ДО из примера 1 нахождение ДД и последующее решение блочно-древовидной задачи с помощью локального алгоритма декомпозиции.

6.1. Выделение древовидной структуры

На рис. 2 приведены результаты применения алгоритма элиминационной игры к нашей задаче. Так как при элиминации переменных новые ребра в графе не добавлялись, то процесс элиминационной игры был эквивалентен поиску симплициальных вершин и соответствующих максимальных клик, которые показаны на рис. 5 со связями между ними. В результате получено дерево клик, к которому можно применить локальный алгоритм декомпозиции. Другим способом на-

хождения дерева клик является поиск максимального стягивающего дерева для двойственного графа (рис.1).

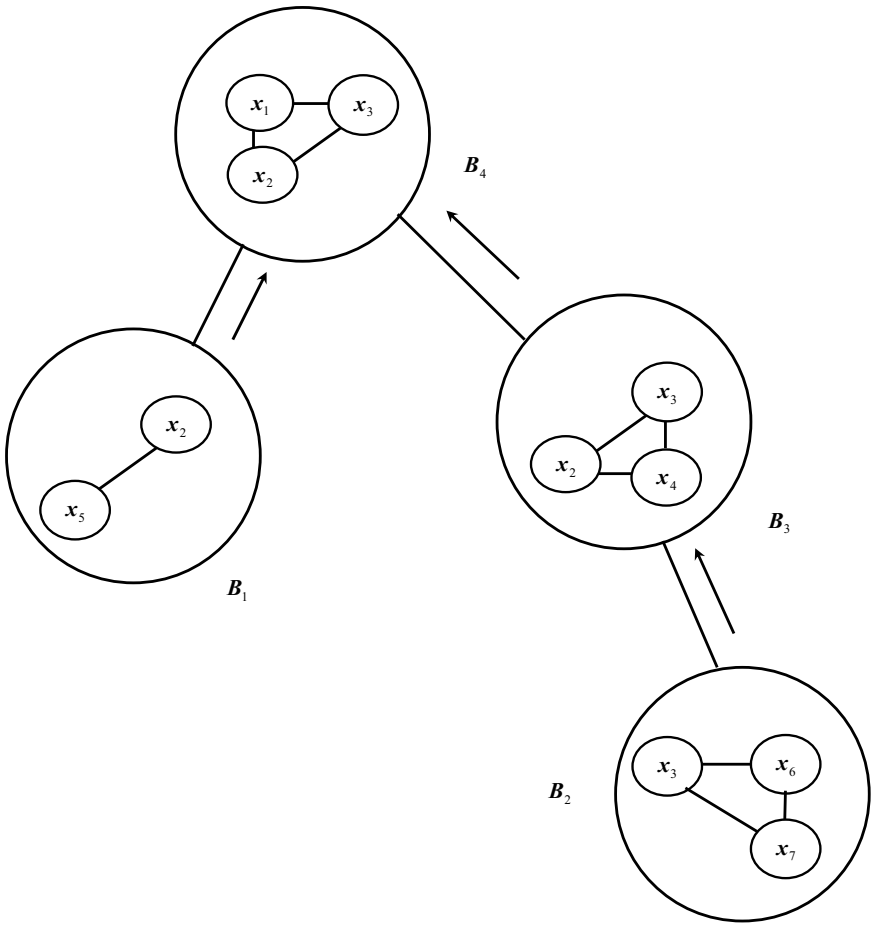


Рис. 5. Древовидная декомпозиция для примера 1

6.2. Применение локального алгоритма декомпозиции

Рассмотрим решение подзадачи, соответствующей блоку B_1. Поскольку этот блок связан с блоком B_4, нам нужно решить задачу ДО с переменными $X_{B_1-B_4}$ для всех возможных наборов переменных $X_{B_1 \bigcap B_4}$. Таким образом, поскольку $X_{B_1-B_4} = \{x_5\}$, а $X_{B_1 \bigcap B_4} = \{x_2\}$, то подзадача имеет вид:

$$h_{B_1}(x_2) = \max_{x_5} \{4x_5\}$$

при ограничениях

$$2x_2 + 3x_5 \leq 4, \quad x_j = 0, 1, \ j \in \{2, 5\}$$

Решение задачи запишем в виде следующей таблицы:

Таблица 1. Вычисление $h_{B_1}(x_2)$

x_2	h_{B_1}	$x_5^*(x_2)$
0	4	1
1	0	0

Следующая подзадача, соответствующая листу дерева клик — блоку B_2, имеет вид:

$$h_{B_2}(x_3) = \max_{x_6,x_7} \{6x_6 + x_7\}$$

при ограничениях

$$2x_3 + 3x_6 + 2x_7 \le 5, \quad x_j = 0, 1, \ j \in \{3, 6, 7\}$$

Решение этой подзадачи:

Таблица 2. Вычисление $h_{B_2}(x_3)$

x_3	h_{B_2}	$x_6^*(x_3)$	$x_7^*(x_3)$
1	1	7	1
1	0	6	1

Подзадача, соответствующая блоку B_3, имеет вид:

$$h_{B_3}(x_2, x_3) = \max_{x_4} \{h_{B_2}(x_3) + 5x_4\}$$

при ограничениях

$$2x_2 + 3x_3 + 3x_4 \le 5, \quad x_j = 0, 1, \ j \in \{2, 3, 4\}$$

Таблица 3. Вычисление $h_{B_3}(x_2, x_3)$

x_2	x_3	h_{B_3}	$x_4^*(x_2, x_3)$
0	0	12	1
0	1	6	0
1	0	12	1
1	1	6	0

Последняя подзадача, имеет вид:

$$h_{B_4} = \max_{x_1,x_2,x_3} \{h_{B_1}(x_2) + h_{B_3}(x_2, x_3) + 2x_1\}$$

при ограничениях

$$3x_1 + 4x_2 + x_3 \le 6, \quad x_j = 0, 1, \ j \in \{1, 2, 3\}$$

Таблица 4. Вычисление h_{B_4}

h_{B_4}	x_1^*	x_2^*	x_3^*
18	1	0	0

Итак, оптимальное значение целевой функции 18. Найдем оптимальные значения переменных, используя информацию в таблицах 1 – 4:
$x_1^* = 1,\ x_2^* = 0,\ x_3^* = 0$ (таблица 4).
Из таблицы 3, учитывая, что $x_2^* = 0,\ x_3^* = 0$, найдем $x_4^* = 1$.
Из таблицы 2, учитывая, что $x_3^* = 0$, найдем $x_6^* = 1, x_7^* = 1$.
Из таблицы 1, учитывая, что $x_2^* = 0$, найдем $x_5^* = 1$.
Итак, найдено оптимальное решение: (1, 0, 0, 1, 1, 1, 1); оптимальное значение ЦФ равно 18.

7. Заключение

Таким образом, локальные алгоритмы декомпозиции задач дискретной оптимизации в сочетании с методами древовидной декомпозиции – перспективный подход, делающий возможными решение прикладных разреженных задач дискретной оптимизации.

Перспективными направлениями дальнейших исследований являются разработка эффективных схем выделения древовидных структур при решении конкретных задач дискретной оптимизации, обладающих специальной структурой, применение постоптимального анализа при решении пакетов задач, получающихся в результате применения локальных алгоритмов декомпозиции, построение многоуровневых схем декомпозиции.

Список цитируемых источников

1. *Евстигнеев В.А., Касьянов В.Н.* Толковый словарь по теории графов в информатике и программировании. — Новосибирск: Наука, 1999. — 288 с.
   (электронная версия словаря доступна по адресу — http://pco.iis.nsk.su/grapp).
2. *Емеличев В.А., Мельников О.И., Сарванов В.И., Тышкевич Р.И.* Лекции по теории графов. — М.: Наука, 1990. — 384 с.
3. *Журавлев Ю.И.* Избранные научные труды. — М.: Магистр, 1998. — 420 с.
4. *Финкельштейн Ю.Ю.* О решении задач дискретного программирования специального вида // Экономика и математические методы. — 1965. — 1, N 2. — P.262–270.
5. *Щербина О.А.* О несериальной модификации локального алгоритма декомпозиции задач дискретной оптимизации // Динамические системы. — 2005. — Вып.19. — С.179—-190.
6. *Щербина О.А.* О локальных алгоритмах решения задач дискретной оптимизации // Проблемы кибернетики. — 1983. — N 40. — P. 171–200.
7. *Amestoy P.R., Davis T.A., Duff I.S.* An approximate minimum degree ordering algorithm // SIAM Journal on Matrix Analysis and Applications. — 1996. — 17, N.4. — P.886—-905.

8. *Arnborg S., Corneil D.G., Proskurowski A.* Complexity of finding embeddings in a k-tree // SIAM J. Alg. Disc. Meth. — 1987. — 8. — P. 277–284.
9. *Arnborg S., Lagergren J., Seese D.* Easy problems for tree-decomposable graphs // J. of Alg. — 1991. — 12. — P.308–340.
10. *Bernstein P.A., Goodman N.* Power of natural semijoins // SIAM J. Comput. — 1981. — 10, N 4. — P.751–771.
11. *Berry A., Blair J., Heggernes P., Peyton B.* Maximum cardinality search for computing minimal triangulations of graphs // Algorithmica. — 2004. — 39. — P. 287—298.
12. *Bertele U., Brioschi F.* Nonserial Dynamic Programming. — New York: Academic Press. — 1972.
13. *Blair J.R.S., Peyton B.W.* An introduction to chordal graphs and clique trees // In J.A.George, J.R.Gilbert, J.W.H.Liu (eds). Sparse Matrix Computations: Graph Theory Issues and Algorithms, Springer Verlag. — 1993. — P.1–30.
14. *Bodlaender H.L.* Discovering Treewidth // Proceedings of SOFSEM Springer–Verlag, LNCS 3381. — 2006. — P. 1—16.
15. *Cook W., Seymour P.D.* An algorithm for the ring-routing problem. — Bellcore technical memorandum. — Bellcore. — 1994.
16. *Cook W., Seymour P.D.* Tour merging via branch-decomposition // INFORMS Journal on Computing. — 2003. — v.15, N3. — P.233–248.
17. *Courcelle B.* Graph rewriting: An algebraic and logic approach // Handbook of Theoretical Computer Science J. Van Leeuwen (ed.). , Volume B. — Elsevier. — 1990. – P.193-242.
18. *Dechter R.* Bucket elimination: A unifying framework for reasoning // Articial Intelligence. — 1999. — 113. — P.41–85.
19. *Dechter R., El Fattah Y.* Topological parameters for time-space tradeoff // Artif. Intell. — 2001. — 125, N 1–2. — P. 93–118.
20. *Dirac G.A.* On rigid circuit graphs // Anh. Math. Sem. Univ. Hamburg. — 1961. — 25. — P.71—76.
21. *Fulkerson D.R., Gross O.A.* Incidence matrices and interval graphs // Pacific Journal of Math. — 1965. — 15. —P.835—855.
22. *Gavril F.* The intersection graphs of subtrees in trees are exactly the chordal graphs // J. Combinatorial Theory Ser.B. — 1974. — 16. — P.47–56.
23. *George J. A., Liu J. W. H.* Computer Solution of Large Sparse Positive Definite Systems. — Englewood Cliffs: Prentice-Hall Inc., 1981.
24. *Gottlob G., Leone N., Scarcello F.* Hypertree decompositions: A survey // In Mathematical foundations of computer science. 26th international symposium, MFCS 2001. Proceedings. — Berlin: Springer. — Lect. Notes Comput. Sci. — 2001. — 2136. —P. 37–57.
25. *Hajnal A., Suranyi J.* Über die Ausflösung von Graphen in vollständige Teilgraphen // Ann. Univ. Sci. Budapest. — 1958. — P. 113–121.
26. *Heggernes P.* Treewidth, partial k-trees, and chordal graphs. – Internet document: http://www.ii.uib.no/ pinar/chordal.pdf
27. *Heggernes P.* Minimal triangulations of graphs: A survey // Discrete Mathematics. — 2006. — 306, 3. — P.297–317.

28. *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. — http://ie.tamu.edu/People/faculty/Hicks/bwtw.pdf
29. *Hooker J.* Logic-based methods for optimization: combining optimization and constraint satisfaction. — John Wiley, 2000. — 495 p.
30. *Koster A.M.C.A., Bodlaender H.L., van Hoesel S. P. M.* Treewidth: Computational experiments // In H. Broersma, U. Faigle, J. Hurink, S. Pickl (eds). Electronic Notes in Discrete Mathematics. — Elsevier Science Publishers. — 2001. — V.8.
31. *Neumaier A., Shcherbina O.* Nonserial dynamic programming and local decomposition algorithms in discrete programming (submitted). Available online: http://www.optimization-online.org/DB_HTML/2006/03/1351.html
32. *Parter S.* The use of linear graphs in Gauss elimination // SIAM Review. — 1961. — 3. — P.119–130.
33. *Robertson N., Seymour P.D.* Graph minors. II. Algorithmic aspects of tree width // J. Algorithms. — 1986. —7. —P.309–322.
34. *Shibata Y.* On the tree representation of chordal graphs // J. Graph Theory. — 1988. — 12. — P.421-428.
35. *Tarjan R.E., Yannakakis M.* Simple linear-time algorithms to test chordality of graphs, test acyclity of hypergraphs, and selectively reduce acyclic hypergraphs // SIAM J. Comput. — 1984. — 13. — P.566–579.
36. *Yannakakis M.* Computing the minimum fill-in is NP-complete // SIAM J. Alg. Disc. Meth. — 1981. — 2. — P.77—79.

Получено 21.10.2006