УДК 519.68

ЭЛИМИНАЦИОННЫЕ АЛГОРИТМЫ ДЕКОМПОЗИЦИИ ЗАДАЧ ДИСКРЕТНОЙ ОПТИМИЗАЦИИ

О.А. Щербина

University of Vienna,
Institut für Mathematik
Nordbergstr. 15, Vienna 1090, Austria
e-mail: *oleg.shcherbina@univie.ac.at*

Abstract

In this paper a class of elimination algorithms of decomposition of discrete optimization problems including local algorithms of decomposition, nonserial dynamic programming algorithms, bucket elimination algorithms, tree decomposition techniques is considered. The review is made and the basic features of the elimination algorithms of decomposition representing rather promising approach allowing in some cases to solve large-scale discrete optimization problems are described.

Введение

Использование моделей и алгоритмов дискретной оптимизации (ДО) позволяет решать многие практические задачи, такие, как задачи теории расписаний, оптимизации на сетях, маршрутизации трафика в коммуникационных сетях, задачи размещения экономических объектов, доказательство теорем, задачи искусственного интеллекта и др. К сожалению, большинство интересных задач ДО являются NP-трудными и решение их в худшем случае может требовать построения дерева поиска решений экспоненциального размера. В настоящее время практически никто не ожидает появления полиномиального алгоритма решения NP-трудных задач [10]. Многие практические задачи ДО содержат огромное число переменных и/или ограничений, что создает сложности при попытке решения этих задач с помощью современных решателей.

Теория сложности доказывает, что универсальность и эффективность являются противоречивыми требованиями к сложности алгоритма.Однако сложность некоторого класса задач ДО снижается, если этот класс может быть разбит на подклассы и специальная структура этих подклассов может быть учтена при построении алгоритма. В последнее время наблюдается потребность в построении и анализе алгоритмов, существенно более быстрых, чем полный перебор, хотя и не полиномиальных (так называемые супер-полиномиальные алгоритмы, с помощью которых находятся оптимальные решения NP-полных задач – см. обзор Woeginger [22]). Перспективной представляется разработка декомпозиционных методов в ДО ([14], [20], [21]), позволяющих справиться с решением NP-трудных задач ДО, что чрезвычайно актуально по вышеизложенным причинам. Обычно прикладные задачи ДО имеют специальную структуру, и матрицы ограничений характеризуются большим количеством нулей (разреженные матрицы). Перспективными декомпозиционными методами, использующими разреженность матрицы ограничений задач ДО представляется класс элиминационных алгоритмов декомпозиции, включающая локальные алгоритмы декомпозиции [2], [4], алгоритмы несериального динамического программирования (НСДП)

([7], [15], [16]), алгоритмы сегментной элиминации [8], методы древовидной декомпозиции [13]. Исторически НСДП появилось чуть позже обычного динамического программирования (ДП), но в настоящее время слабо известно среди специалистов в области теории оптимизации, особенно в отечественной научной литературе. Связь НСДП с локальными алгоритмами ([2], [4], [3]) обусловлена тем, что, как и локальные алгоритмы, НСДП решает задачи, преобразуя локальную информации в глобальную. Интересные аналогии прослеживаются между НСДП и методом исключения переменных Гаусса при решении систем линейных переменных. Одной из общих для этих методов графовых интерпретаций является *элиминационная игра* (Parter [17]).

В настоящей статье рассмотрен класс элиминационных алгоритмов декомпозиции задач дискретной оптимизации, включающий локальные алгоритмы декомпозиции, алгоритмы несериального динамического программирования, алгоритмы сегментной элиминации, методы древовидной декомпозиции. Сделан обзор и описаны основные черты элиминационных алгоритмов декомпозиции, представляющих собой весьма перспективный подход, позволяющий в ряде случаев решать задачи дискретной оптимизации большой размерности.

1. Сериальная и несериальная задачи дискретной оптимизации и их графовые представления

Согласно [7], динамическое программирование (ДП) и НСДП, в частности, рассматривается как общий подход (парадигма) к решению задач, основанный на декомпозиции исходной задачи на меньшие подзадачи. ДП, известное также как метод элиминации переменных, исключает (или элиминирует) переменные в определенном порядке и выводящее новые зависимости (новые компоненты целевой функции – в случае задачи оптимизации) среди оставшихся переменных.

Напомним, что **сериальной** (т.е. последовательной) задачей ДП без ограничений называется следующая задача:

$$\min_X f(X) = \min_X \sum_{i \in K} f_i(X^i),$$

где $X = \{x_1, \ldots, x_n\}$ - множество дискретных переменных,

$$K = \{1, 2, \ldots, n-1\}\,;\ X^i = \{x_i, x_{i+1}\}\,.$$

Функция $f(X)$ называется целевой функцией (ЦФ), а функции $f_i(X^i)$ - компонентами ЦФ и могут задаваться таблично. На рис.1 показан граф взаимосвязей этой задачи.

Согласно [7], несериальная задача оптимизации без ограничений имеет вид:

$$\min_X f(X) = \min_X \sum_{i \in K} f_i(X^i),$$

где $X = \{x_1, \ldots, x_n\}$ – множество дискретных переменных,

$$K = \{1, 2, \ldots, t\}\,;\ X^i \subset X.$$

Здесь t – число компонентов ЦФ.

Определение 1. [7]. Две переменные $x \in X$ и $y \in Y$ взаимосвязаны в несериальной задаче оптимизации без ограничений, если существует такой компонент ЦФ $f_k(X^k)$, что обе переменные x и y принадлежат X^k (иногда говорят, что переменные x и y являются соседними или смежными по компоненту $f_k(X^k)$).

Рассмотрим несериальную задачу ДО с ограничениями следующего вида:

$$F(x_1, x_2, \ldots, x_n) = \sum_{k \in K} f_k(Y^k) \to \max \tag{1}$$

при ограничениях

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

$$x_j \in D_j,\ j \in \{1, \ldots, n\}, \tag{3}$$

где

$$Y^k \subseteq \{x_1, x_2, \ldots, x_n\},\ k \in K = \{1, 2, \ldots, t\}; \tag{4}$$

$$X^i \subseteq \{x_1, x_2, \ldots, x_n\},\ R_i \in \{\leq, =, \geq\},\ i \in M = \{1, 2, \ldots, m\}. \tag{5}$$

Определение 2. [7]. Две переменные $x \in X$ и $y \in Y$ взаимосвязаны в несериальной задаче оптимизации с ограничениями, если они появляются вместе в одном 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 (интерпретацию) несериальной задачи оптимизации, что позволит в дальнейшем использовать ДП для решения этой задачи. Нам потребуется понятие графа взаимосвязей (interaction graph [7]), естественным образом представляющего структуру задачи ДО.

Определение 3. Графом взаимосвязей несериальной задачи оптимизации (без ограничений или с ограничениями) называется неориентированный граф $G = (V, X)$, для которого

1. множество вершин X графа соответствует множеству переменных задачи оптимизации;
2. две вершины графа смежны тогда и только тогда, когда соответствующие им переменные взаимосвязаны.

Определение 4. Множество переменных, взаимосвязанных с переменной $x \in X$, обозначается $Nb(x)$ и называется **окрестностью** переменной x.

2. Процедура элиминации переменных для задач оптимизации без ограничений

Сериальная задача без ограничений

В сериальной задаче ДО вначале рассмотрим переменную x_1. Поскольку лишь компонент $f_1(X^1) = f_1(x_1, x_2)$ зависит от x_1, для минимизации $f(X)$ по отношению

Рис. 1. Граф взаимосвязей переменных в сериальной задаче ДО

к x_1 достаточно вычислить

$$\min_{\{x_1\}} f_1(x_1, x_2) = h_1(x_2)$$

для всех возможных значений x_2, так как x_1 взаимосвязан лишь с x_2, и запомнить оптимальные значения $x_1^*(x_2)$. Функция $f_1(x_1, x_2)$ заменяется таблично заданной функцией $h_1(x_2)$, которая становится новым компонентом ЦФ. В результате выполнения описанной операции, называемой **элиминацией** переменной x_1 [7], задача принимает вид:

$$\min_{X-\{x_1\}} \left[h_1(x_2) + \sum_{i \in K-\{1\}} f_i(X^i) \right].$$

Далее производится элиминация переменной x_2 из этой новой задачи. Поскольку переменная x_2 взаимосвязана лишь с x_3, то вычисляется оптимальное значение x_2 при всех возможных значений x_3 и в результате запоминается функция $h_2(x_3)$ и оптимальные значения $x_2^*(x_3)$.

Применяя далее описанную процедуру, последовательно производя элиминацию переменных $x_3, x_4, \ldots, x_n$, в конце концов получим оптимальное значение $f(X)$. Оптимальное решение X может быть найдено с помощью обратного шага процедуры ДП, который определяет последовательно $x_n^*, x_{n-1}^*, \ldots, x_1^*$ из сохраненных таблиц значений $x_n^*, x_{n-1}^*(x_n), \ldots, x_1^*(x_2)$.

Несериальная задача без ограничений

Рассмотрим какое-либо упорядочение (одно из $n!$ возможных) переменных из множества X

$$\alpha = \{y_1, \ldots, y_n\}.$$

Для этого упорядочения задача может быть решена с помощью описанной выше процедуры ДП, которую можно разбить на две части.

а. Прямая часть

Вначале рассмотрим переменную y_1. Тогда

$$\min_X f(X) = \min_{X-\{y_1\}} \min_{\{y_1\}} \sum_{i \in K} f(X^i) = \min_{X-\{y_1\}} \left(\sum_{i \in K-K_1} f_i(X^i) + \min_{\{y_1\}} \sum_{i \in K_1} f_i(X^i) \right) =$$

$$= \min_{X-\{y_1\}} \left(\sum_{i \in K-K_1} f_i(X^i) + h_1(Nb(y_1)) \right),$$

где

$$K_1 = \{i : X^i \cap \{y_1\} \neq \emptyset\}.$$

Первый шаг оптимизационной процедуры состоит из вычисления с помощью полного перебора

$$\min_{\{y_1\}} \sum_{i \in K_1} f_i(X^i) = h_1(Nb(y_1))$$

и запоминания оптимальных частичных решений y_1 как функции $Nb(y_1)$, а именно, $y_1^*(Nb(y_1))$.

Минимизацию $f(X)$ по отношению к переменной y_1, для всех возможных наборов $Nb(y_1)$, будем называть **элиминацией** переменной y_1. После элиминации y_1 задача имеет вид:

$$\min_{X-\{y_1\}} \left(\sum_{i \in K-K_1} f_i\left(X^i\right) + h_1\left(Nb(y_1)\right) \right),$$

т.е. имеет тот же вид, что и исходная задача, причем функция $h_1(Nb(y_1))$ может быть рассмотрена как компонент новой ЦФ. Далее та же процедура может быть применена для поочередной элиминации переменных $y_2, \ldots, y_n$. На каждом шаге j запоминаются новый компонент h_j и оптимальные наборы y_j^* как функции множества переменных $Nb(y_j \mid y_1, \ldots, y_{j-1})$, взаимосвязанных с переменной y_j в текущей задаче, полученной из исходной задачи путем элиминации $y_1, \ldots, y_{j-1}$. Так как множество $Nb(y_n \mid y_1, \ldots, y_{n-1})$ пусто, то элиминация y_n дает оптимальное значение $f(X)$.

В. Обратная часть.

Эта часть процедуры состоит в последовательном определении $y_n^*, y_{n-1}^*, \ldots, y_1^*$ - оптимальных наборов из заполненных таблиц $y_1^*(Nb(y_1)), y_2^*(Nb(y_2 \mid y_1)), \ldots, y_n^*$.

В [1] приведен пример решения несериальной задачи ДО без ограничений.

3. Процедура элиминации переменных для задач дискретной оптимизации с ограничениями

Рассмотрим задачу ДО с ограничениями и предположим без потери общности, что порядок элиминации переменных следующий: $x_1, \ldots, x_n$. Опишем процедуру элиминации для переменной x_1. Переменная x_1 входит в подмножество K_1 компонентов ЦФ:

$$K_1 = \left\{k \mid x_1 \in Y^k\right\}$$

и в подмножество ограничений U_1:

$$U_1 = \left\{i \mid x_1 \in X^i\right\}.$$

Одновременно с x_1 в компоненты ЦФ Y^k, $k \in K_1$ и в ограничения U_1 входят переменные из окрестности $Nb(x_1)$.

Переменной x_1 соответствует следующая подзадача P_1 исходной задачи ДО:

$$h_1(Nb(x_1) = \max_{x_1} \left\{ \sum_{k \in K_1} f_k(Y^k) \mid g_i(X^i \ R_i \ 0, \ i \in U_1, \ x_j \in D_j, \ x_j \in Nb(x_1) \right\}.$$

Исходная задача ДО может быть преобразована следующим образом:

$$\max_{x_1,\ldots,x_n} \left\{ \sum f_k(Y^k) \mid g_i(X^i \ R_i \ 0, \ i \in M, \ x_j \in D_j, \ j \in N \right\} =$$

$$= \max_{x_1,\ldots,x_{n-1}} \left\{ \sum_{k \in K - K_1} f_k(Y^k) + h_1(Nb(x_1) \mid g_i(X^i) \ R_i \ 0, \ i \in M - U_1, \ x_j \in D_j, \ j \in X - \{x_1\} \right\}.$$

В новой задаче $n-1$ переменная; по сравнению с исходной задачей в ней исключены ограничения с индексами из U_1 и компоненты ЦФ $\sum_{k \in K_1} f_k(Y^k)$, но появился новый компонент ЦФ $h_1(Nb(x_1))$.

Процедура элиминации элиминирует оставшиеся переменные одну за другой аналогичным образом. При этом, подобно выше описанным процедурам элиминации, необходимо запоминать таблицы с оптимальными частичными решениями на каждом шаге процесса.

На шаге n описанного процесса мы элиминируем переменную x_n и находим оптимальное значение ЦФ. Затем нужно выполнить обратную часть процедуры элиминации для нахождения оптимального решения. Решение задачи ДО с помощью НСДП подробно описано в [3]. Ниже рассмотрено решение несериальной задачи ДО с ограничениями с помощью сегментной элиминации.

В качестве примера несериальной задачи с ограничениями рассмотрим задачу целочисленного линейного программирования:

Пример 1.

$$\begin{array}{rcrcrcrcrcrcrll}
2x_1 & + & 3x_2 & + & x_3 & + & 5x_4 & + & 4x_5 & + & 6x_6 & + & x_7 & \to \max & \\
3x_1 & + & 4x_2 & + & x_3 & & & & & & & & & \leqslant 6, & (C_1) \\
 & & 2x_2 & + & 3x_3 & + & 3x_4 & & & & & & & \leqslant 5, & (C_2) \\
 & & 2x_2 & & & & & + & 3x_5 & & & & & \leqslant 4, & (C_3) \\
 & & & & 2x_3 & & & & & + & 3x_6 & + & 2x_7 & \leqslant 5, & (C_4)
\end{array}$$

$x_j = 0, \ 1, \ j = 1, \ldots, 7.$

4. Элиминационный процесс и элиминационная игра Партера

Рассмотрим неориентированный граф $G = (V, E)$ и вершину $y \in V$.

Определение 5. [7] Граф, полученный из графа $G = (V, E)$ с помощью

- удаления вершины y и всех ребер $O(y)$, исходящих из нее и
- соединения всех ранее несмежных вершин в $Nb(y)$,

а именно, граф $(V - \{y\}, (E - O(y)) \bigcup I^*(Nb(y)))$, называется *$y$-элиминационным графом* графа G и обозначается G_y. Описанная операция называется *элиминацией вершины y.*

Здесь $I^*(Y)$ - множество ребер полного графа с множеством вершин Y.

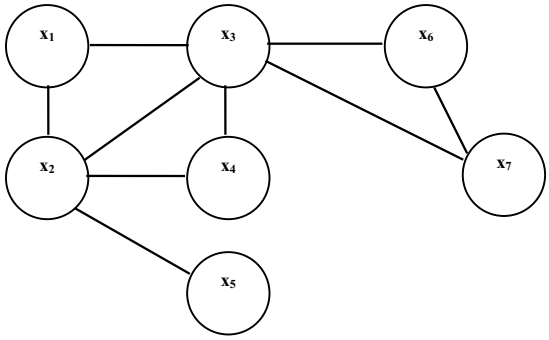

Рис. 2. Граф взаимосвязей переменных в несериальной задаче ДО с ограничениями из примера 1.

Для данного упорядочения $\alpha = (y_1, y_2, \ldots, y_n)$ элиминационный процесс состоит в последовательной элиминации вершин $y_1, y_2, \ldots, y_n$ в текущем графе. Этот процесс порождает последовательность графов:

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

так что G^j – y_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 (Теорема об инвариантности)**.** [7]. *Любой граф, полученный из графа $G = (V, E)$ с помощью поочередной элиминации всех вершин из некоторого подмножества $Y \subset V$, не зависит от порядка элиминации вершин.*

Определение 6. Граф, полученный из G с помощью элиминации всех вершин Y в некотором порядке, называется Y-элиминационным графом графа G и обозначается G_Y; эта операция называется элиминацией подмножества Y.

Теорема 2. *(Финальная теорема)* [7]. *Если $G = (V, E)$ – граф, Σ – полное в G подмножество вершин V, тогда существует оптимальное упорядочение, в котором вершины Σ по порядку являются последними.*

Процесс преобразования первичного графа взаимодействия переменных, соответствующий процедуре НСДП, известен как *элиминационная игра*, которая впервые была введена Партером [17], 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). Процесс преобразования первичного графа взаимосвязей, соответствующий процедуре НСДП, известен как *«элиминационная игра»*, которая впервые была введена Партером [17],

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 [11] 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.

Различные пополненные графы получаются в результате обработки вершин графа G в различном порядке. Поэтому для получения небольшого пополнения перед использованием элиминационной игры необходимо найти хороший порядок вершин данного графа. Отметим, что задача нахождения упорядочения, обеспечивающего минимальное пополнение, является NP–трудной [24].

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 популярных эвристик является эвристика «Minimum Degree», которая выбирает на каждом шаге i вершину v с минимальной степенью в G^{i-1} .

5. Блочная элиминация переменных

В ряде случаев целесообразно использовать в процедуре НСДП так называемую "блочную элиминацию" ([7], [2], [4]), которая осуществляет элиминацию нескольких вершин за один раз. Рассмотрим упорядоченное разбиение переменных множества X на блоки:

$$\Psi = (Y_1, \ldots, Y_m),\ m \leq n.$$

Для этого упорядоченного разбиения задача оптимизации может быть решена с помощью НСДП.

А. Прямая часть

Рассмотрим вначале блок Y_1. Тогда

$$\min_X f(X) = \min_{X-Y_1} \min_{Y_1} \sum_{i \in K} f(X^i) = \min_{X-Y_1} \left(\sum_{i \in K-K_1} f_i(X^i) + \min_{Y_1} \sum_{i \in K_1} f_i(X^i) \right) =$$

$$= \min_{X-Y_1} \left(\sum_{i \in K-K_1} f_i(X^i) + h_1(Nb(Y_1)) \right),$$

где

$$K_1 = \{i : X^i \cap Y_1 \neq \emptyset\}.$$

Первый шаг процедуры оптимизации состоит в вычислении функции

$$h_1(Nb(Y_1)) = \min_{Y_1} \sum_{i \in K_1} f_i(X^i),$$

используя полный перебор и запоминая оптимальные наборы Y_1 как функцию переменных из окрестности $Nb(Y_1)$, а именно $Y_1^*(Nb(Y_1))$.

На втором шаге производится минимизация $f(X)$ по всем возможным наборам $Nb(Y_1)$ называется элиминацией или исключением блока Y_1. Задача оптимизации, оставшаяся после исключения Y_1, выглядит так:

$$\min_{X-Y_1} \left(\sum_{i \in K-K_1} f_i(X^i) + h_1(Nb(Y_1)) \right).$$

Отметим, что она имеет тот же вид, что и исходная задача, причем функция $h_1(Nb(Y_1))$ может быть рассмотрена как компонент новой ЦФ. Далее та же процедура может быть применена для поочередной элиминации блоков $Y_2, \ldots, Y_m$. На каждом шаге j запоминаются новая компонента h_j и оптимальные частичные решения Y_j^* как функции $Nb(Y_j \mid Y_1, \ldots, Y_{j-1})$ множества переменных, взаимодействующих, по крайней мере, с одной переменной Y_j в текущей задаче, полученной из исходной задачи путем элиминации $Y_1, \ldots, Y_{j-1}$. Так как множество $Nb(Y_m \mid Y_1, \ldots, Y_{m-1})$ пусто, то элиминация Y_m дает оптимальное значение $f(X)$.

В. Обратная часть.

Эта часть процедуры состоит в последовательном определении $Y_m^*, Y_{m-1}^*, \ldots, Y_1^*$ - оптимальных наборов из заполненных таблиц $Y_1^*(Nb(Y_1)), Y_2^*(Nb(Y_2 \mid Y_1)), \ldots, Y_m^*$.

Одним из возможных способов выбора множеств Y^j является определение их в виде множеств "неотличимых" переменных [5] (две переменные x_i и x_j называются *неотличимыми*, если $Nb(x_i) = Nb(x_j)$). Тогда, если переменная x_i отобрана для элиминации, то x_j может быть отобрана также без порождения дополнительного пополнения. Целесообразно выбрать для элиминации x_i и x_j вместе, или "блоком". В этом случае возможно ввести новую блочную переменную $\{x_i, x_j\}$ (в [5] она названа *супер-переменной*). Понятно, что подобные супер-переменные содержатся в одних и тех же ограничениях.

В качестве другой возможности можно рассмотреть переменные x_i, x_j с вложенными окрестностями (т.е. если $Nb(x_j) \subseteq Nb(x_i)$). В этом случае мы можем рассматривать эти 2 переменные вместе, "блоком". Локальный алгоритм декомпозиции, предложенный в [4], фактически реализует алгоритм блочной элиминации.

Рассмотрим неориентированный граф $G = (V, E)$; $Y \subset V$.

Определение 7. Граф, полученный из графа G с помощью

- удаления вершин из Y и всех ребер $I(Y) \cup \mathrm{O}(Y)$, где $\mathrm{O}(Y)$ - множество ребер, соединяющих вершины Y и $V - Y$, и
- соединения всех ранее несмежных вершин в $Nb(Y)$,

а именно, граф $(V - Y, I(V - Y) \bigcup I^*(Nb(Y)))$, называется *$Y$–блочно-элиминационным графом* графа G и обозначается G^Y. Описанная операция называется **блочной элиминацией** множества Y.

6. Метод сегментной элиминации

Алгоритм сегментной элиминации (bucket elimination), предложен Dechter [8] для упорядочения процесса вычислений элиминационных алгоритмов. Здесь этот алгоритм переформулирован для решения задач ДО.

Алгоритм сегментной элиминации использует в качестве входной информации упорядоченное множество переменных и множество ограничений. Каждой переменной x_j ставится в соответствие сегмент (подмножество) $\Sigma^{(x_j)}$ ограничений и компонентов ЦФ, построенный следующим образом: все ограничения и компоненты ЦФ, включающие переменную x_j, и не включающие переменных с бóльшим индексом (согласно заданному упорядочению переменных), помещаются в сегмент $\Sigma^{(x_j)}$. После того, как все сегменты построены, алгоритм элиминации начинает проходить их с конца до начала. Алгоритм находит новые компоненты ЦФ, применяя так называемый «оператор элиминации» (в нашем случае – решая соответствующие подзадачи оптимизации) ко всем ограничениям и компонентам ЦФ рассматриваемого сегмента. Новые компоненты ЦФ, учитывающие влияние переменной x_j на оставшуюся часть задачи, помещаются в соответствующие сегменты ниже.

Рассмотрим применение метода сегментной элиминации для несериальной задачи ДО с ограничениями, описанной выше в примере 1. Для решения используем порядок элиминации переменных $\alpha = \{x_5, x_1, x_2, x_4, x_3, x_6, x_7\}$, полученный с помощью эвристики «Minimum Degree». Строим сегменты (подмножества ограничений), начиная с последней (согласно порядку α) переменной x_7. В сегмент $\Sigma^{(x_7)}$ включаем все ограничения задачи ДО, содержащие переменную x_7, т.е. сегмент $\Sigma^{(x_7)}$ состоит из ограничения C_4:

$$\Sigma^{(x_7)} = \{C_4\}.$$

Далее ограничение C_4 из рассмотрения исключается. Аналогично получаем: $\Sigma^{(x_6)} = \emptyset$, $\Sigma^{(x_3)} = \{C_1, C_2\}$, $\Sigma^{(x_4)} = \emptyset$, $\Sigma^{(x_2)} = \{C_3\}$, $\Sigma^{(x_1)} = \emptyset$, $\Sigma^{(x_5)} = \emptyset$.

Решаем подзадачу оптимизации, соответствующую сегменту $\Sigma^{(x_7)}$:
Для каждого набора значений x_3 и x_6, вычислить значение x_7, для которого:

$$h_7(x_3, x_6) = \max_{x_7}\{x_7 \mid 2x_3 + 3x_6 + 2x_7 \leq 5, x_j \in \{0, 1\}\}.$$

Полученную функцию $h_7(x_3, x_6)$ помещаем в сегмент $\Sigma^{(x_6)}$, рассматриваем для этого сегмента задачу

$$h_6(x_3) = \max_{x_6}\{6x_6 + h_7(x_3, x_6) \mid x_j \in \{0, 1\}\}$$

и строим таблицу 2.

Таблица 1. Вычисление $h_7(x_3, x_6)$

x_3	x_6	h_7	x_7^*
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

Таблица 2. Вычисление $h_6(x_3)$

x_3	h_6	x_6^*
0	7	1
1	6	1

Полученную функцию $h_6(x_3)$ помещаем в сегмент $\Sigma^{(x_3)}$ и решаем соответствующую задачу

$$h_3(x_1,\, x_2,\, x_3) = \max_{x_3}[\, x_3 + h_6(x_3)\,]$$

$$\begin{aligned} 3x_1 \;+\; 4x_2 \;+\; x_3 &\leqslant 6,\\ 2x_2 \;+\; 3x_3 \;+\; 3x_4 &\leqslant 5,\\ x_j = 0,\, 1,\ j = 1,2,3,4.& \end{aligned}$$

Таблица 3. Вычисление $h_3(x_1, x_2, x_4)$

x_1	x_2	x_4	h_3	x_3^*
0	0	0	7	1
0	0	1	7	0
0	1	0	7	1
0	1	1	7	0
1	0	0	7	1
1	0	1	7	0
1	1	0	–	–
1	1	1	–	–

Функцию $h_3(x_1, x_2, x_4)$ запишем в сегмент $\Sigma^{(x_4)}$ и решим задачу

$$h_4(x_1, x_2) = \max_{x_4}\{5x_4 + h_3(x_1, x_2, x_4) \mid x_j \in \{0, 1\}\}.$$

Получим таблицу:
Функцию $h_4(x_1, x_2)$ запишем в сегмент $\Sigma^{(x_2)}$ и решим задачу

$$h_2(x_1, x_5) = \max_{x_2}\{3x_2 + h_4(x_1, x_2) \mid 2x_2 + 3x_5 \leq 4, x_j \in \{0, 1\}\}$$

Таблица 4. Вычисление $h_4(x_1, x_2)$

x_1	x_2	h_4	x_4^*
0	0	12	1
0	1	12	1
1	0	12	1
1	1	–	–

Таблица 5. Вычисление $h_2(x_1, x_5)$

x_1	x_5	h_2	x_2^*
0	0	15	1
0	1	12	0
1	0	12	0
1	1	12	0

Полученную функцию $h_2(x_1, x_5)$ помещаем в сегмент $\Sigma^{(x_1)}$ и рассматриваем соответствующую задачу

$$h_1(x_5) = \max_{x_1}\{2x_1 + h_2(x_1, x_5) \mid x_j \in \{0, 1\}\}.$$

Таблица 6. Вычисление $h_1(x_5)$

x_5	h_1	x_1^*
0	15	0
1	14	1

Переместим h_1 в последний сегмент $\Sigma^{(x_5)}$ и решим задачу

$$h_5 = \max_{x_5}\{4x_5 + h_1(x_5) \mid x_j \in \{0, 1\}\}.$$

Получим: $h_5 = 18,\ x_5^* = 1$. Итак, оптимальное значение ЦФ равно 18. Перейдем к обратному шагу процедуры, следуя по сегментам в обратном порядке, и найдем оптимальные значения переменных.
Итак, $x_5^* = 1$. В таблице 1 для h_1, найдем для $x_5 = 1$ соответствующее значение $x_1^* = 1$. Далее в таблице 2 для h_2: $x_1 = 1,\ x_5 = 1$ соответствует $x_2^* = 0$.
Рассмотрим таблицу 3 для h_4: $x_1 = 1,\ x_2 = 0 \Rightarrow x_4^* = 1$.
Таблица 4 для h_3: $x_1 = 1,\ x_2 = 0,\ x_4 = 1 \Rightarrow x_3^* = 0$.
Таблица 5 для h_6: $x_3 = 0 \Rightarrow x_6^* = 1$.
Таблица 6 для h_7: $x_3 = 0,\ x_6 = 1 \Rightarrow x_7^* = 1$.

Итак, найдено оптимальное решение: (1, 0, 0, 1, 1, 1, 1), оптимальное значение ЦФ равно 18.

7. Древовидная декомпозиция задач дискретной оптимизации

В связи с отсутствием циклов в деревьях, многие задачи оптимизации, которые являются трудными на общих графах, легко решаются на деревьях [23]. При решении задачи на дереве обычно возможно найти «локальные» или частичные решения для поддерева. Эти локальные решения можно далее использовать при решении задачи ДО с помощью ДП и, в частности, с помощью локальных алгоритмов декомпозиции. ДП начинает с листьев дерева и переходит от меньших к большим подзадачам (соответствующим поддеревьям). В связи с этим понятна перспективность поиска и выявления древовидных структур на графах. Это может быть осуществлено с использованием методов древовидной декомпозиции [18], использующих реструктуризацию задачи ДО, основанную на выделении древовидной структуры в графе взаимосвязей задачи. При этом вершины древовидной структуры включают множества переменных исходной структуры, каждое из которых рассматривается как единая вершина древовидной структуры. Следует отметить, что после выделения древовидной структуры в задачах ДО применяются локальные алгоритмы декомпозиции, представляющие собой по сути алгоритмы блочной элиминации, рассмотренные выше [3].

Используя динамическое программирование на древовидных декомпозициях с ограниченной древовидной шириной, трудные оптимизационные задачи на графах часто могут быть решены за полиномиальное время [6].

Заключение

Элиминационные алгоритмы декомпозиции задач дискретной оптимизации — перспективный подход, делающий возможным решение практических разреженных задач дискретной оптимизации.

Перспективными направлениями дальнейших исследований являются разработка эффективных схем несериального динамического программирования при решении конкретных задач дискретной оптимизации, обладающих специальной структурой, применение постоптимального анализа при решении пакетов задач, получающихся в результате применения блочной элиминационной схемы, построение многоуровневых элиминационных схем.

Список литературы

1. *Рыжаков А.Н., Щербина О.А., Никольский В.Н.* Математическое программирование. – Симферополь: Крымский институт бизнеса. – 2005. – 264 с.
2. *Финкельштейн Ю.Ю.* О решении задач дискретного программирования специального вида // Экономика и математические методы. – 1965. – 1, N 2. – Р. 262–270.
3. *Щербина О.А.* О несериальной модификации локального алгоритма декомпозиции задач дискретной оптимизации // Динамические системы. – 2005. – Вып.19. – С. 179—190.

4. *Щербина О.А.* О локальных алгоритмах решения задач дискретной оптимизации // Проблемы кибернетики. – 1983. – N 40. – P. 171-200.
5. *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.
6. *Arnborg S., Lagergren J.,Seese D.* Easy problems for tree-decomposable graphs // J. of Alg. – 1991. – 12. – P. 308-340.
7. *Bertele U., Brioschi F.* Nonserial Dynamic Programming. – New York: Academic Press. – 1972.
8. *Dechter R.* Bucket elimination: A unifying framework for reasoning // Articial Intelligence. – 1999. – 113. – P. 41-85.
9. *Dechter R., El Fattah Y.* Topological parameters for time-space tradeoff // Articial Intelligence. – 2001. – 125, No. 1–2, P. 93-118.
10. *Fomin F. V., Todinca I., Kratsch D.* Exact (exponential) algorithms for treewidth and minimum fill-in. – Report N 268. – Bergen: University of Bergen, Department of Informatics. – 2004.
11. *Fulkerson D.R., Gross O.A.* Incidence matrices and interval graphs // Pacific Journal of Math. – 1965. – 15. – P. 835—855.
12. *Heggernes P.* Minimal triangulations of graphs: A survey // Discrete Mathematics. – 2006. – 306, 3. – P. 297-317.
13. *Hicks I.V., Koster A.M.C.A., Kolotoglu E.* Branch and Tree Decomposition Techniques for Discrete Optimization // In: Tutorials in Operations Research. – New Orleans: INFORMS – 2005. – P. 1-29. (http://ie.tamu.edu/People/faculty/Hicks/bwtw.pdf)
14. *Nowak I.* Lagrangian decomposition of block-separable mixed-integer all-quadratic programs // Math. Programming. — 2005. – 102, N 2. P. 295-312.
15. *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
16. *Rosenthal A.* Dynamic programming is optimal for nonserial optimization problems // SIAM J. Comput. – 1982. – V.11, N 1. – P. 47-59.
17. *Parter S.* The use of linear graphs in Gauss elimination // SIAM Review. – 1961. – 3. – P. 119–130.
18. *Robertson N., Seymour P.D.* Graph minors. II. Algorithmic aspects of tree width // J.Algorithms. – 1986. – 7. – P. 309-322.
19. *Shcherbina O.A.* Nonserial Dynamic Programming and Tree Decomposition in Discrete Optimization // Applied Optimization and Metaheuristic Innovations (Abstracts of Int.Conference, Yalta, July 19—-21, 2006). — 2006. — P. 45-46.
20. *Vanderbeck F., Savelsbergh M.W.P.* A Generic View of Dantzig-Wolfe Decomposition for Integer Programming // Operations Research Letters. – 2006. – 34, P. 296-306.
21. *Van Roy T.J.* Cross decomposition for mixed integer programming with applications to facility location. // In: J.P. Brans (ed.), Operations Research. – 1981. – 81. – Amsterdam: North-Holland. — P. 579-587.
22. *Woeginger G.J.* Exact algorithms for NP-hard problems: A survey // In: M. Juenger, G. Reinelt and G. Rinaldi (eds.) «Combinatorial Optimization - Eureka! You shrink!». – LNCS 2570. – Springer. – 2003. – P. 185-207.
23. *Wolle T.* Computational aspects of treewidth, lower bounds and networks reliability. – Dissertation: UU Universiteit Utrecht. – 2005. – 158 pp.
http://igitur-archive.library.uu.nl/dissertations/2005-0614-200103/index.htm
24. *Yannakakis M.* Computing the minimum fill-in is NP-complete // SIAM J. Alg. Disc. Meth. — 1981. — 2. — P. 77–79.