Моделирование и анализ данных
2020. Том 10. № 3. С. 60–68
DOI: https://doi.org/10.17759/mda.2020100305
ISSN: 2219-3758 (печатный)
ISSN: 2311-9454 (online)

Modelling and Data Analysis
2020. Vol 10. no. 3. pp. 60–68
DOI: https://doi.org/10.17759/mda.2020100305
ISSN: 2219-3758 (print)
ISSN: 2311-9454 (online)

МЕТОДЫ ОПТИМИЗАЦИИ

УДК 519.85

Практическая реализация алгоритма декомпозиции путей ориентированного графа

Золотарев И.А.*
Московский авиационный институт (НИУ МАИ),
г. Москва, Российская Федерация
ORCID: https://orcid.org/0000–0002–6437–2212
e-mail: yngvar.antonsson@gmail.com

Рассказова В.А.**
Московский авиационный институт (НИУ МАИ),
г. Москва, Российская Федерация,
ORCID: https://orcid.org/0000–0003–4943–3133
e-mail: varvara.rasskazova@mail.ru

Работа направлена на прояснение некоторых особенностей программной реализации алгоритма декомпозиции путей ориентированного графа. Разобраны алгоритмы для формирования таблицы M для декомпозиции множества путей, сортировки таблицы M по полю N_Σ и расчета балансов. На основе данных алгоритмов и исходного алгоритма декомпозиции путей ориентированного графа разработан комплекс программ на языке программирования Python. Проведены расчеты для случайного графа размерности 100 вершин и приводится время работы предложенного алгоритма. Полученные результаты могут быть использованы при решении задачи организации грузовых железнодорожных перевозок на этапе назначения и перемещения локомотивов. Научная и практическая новизна работы заключается в существенном снижении размерности исходной задачи, что особенно важно в условиях транспортных сетей сложной топологии.

Ключевые слова: теория оптимизации, оптимизация на графах, алгоритм декомпозиции путей ориентированного графа, сильно связный граф, подграфы.

Для цитаты:
Золотарев И.А., Рассказова В.А. Практическая реализация алгоритма декомпозиции путей ориентированного графа // Моделирование и анализ данных. 2020. Том 10. № 3. С. 60–68. DOI: https://doi.org/10.17759/mda.2020100305

***Золотарев Игорь Антонович,** студент магистратуры, Московский авиационный институт, (НИУ МАИ), г. Москва, Российская Федерация, ORCID: https://orcid.org/0000–0002–6437–2212, e-mail: yngvar.antonsson@gmail.com

****Рассказова Варвара Андреевна,** кандидат физико-математических наук, доцент кафедры 804 «Теория вероятностей и компьютерное моделирование», Московский авиационный институт, (НИУ МАИ), г. Москва, Российская Федерация, ORCID: https://orcid.org/0000–0003–4943–3133, e-mail: varvara.rasskazova@mail.ru

ВВЕДЕНИЕ

Практическая реализация алгоритма декомпозиции путей ориентированного графа вызывает несколько побочных задач, требующих решения. Во-первых, нужно построить таблицу, которая будет использоваться в алгоритме. Ручное составление таблицы и ручная обработка данных требует колоссальных временных затрат, особенно для графов большой размерности, пути в которых, как правило, могут достигать нескольких сотен (и даже тысяч) ребер (вершин). Таким образом, совершенно очевидна необходимость частичной автоматизации вышеописанного процесса. Составление таблицы по исходному графу и множеству путей – далеко не тривиальная проблема, рассмотрение которой выходит за пределы данной статьи, поэтому будем полагать, что на вход алгоритма подаются набор путей, а также таблица, в которой указана принадлежность ребер графа к определенным подграфам в виде списка.

Второй задачей, требующей решения в процессе реализации алгоритма, является использование в алгоритме нескольких эвристических критериев, таких как расчет балансов и сортировка по N_{Σ}.

ПОДГОТОВКА ТАБЛИЦЫ ДЛЯ ДЕКОМПОЗИЦИИ

Пусть задан ориентированный граф $\vec{G}$ – (V,E) и набор подграфов G_1, G_2 ... G_k, и пусть $P = \{p_1, \dots, p_m\}$ – заданный набор путей графа $\vec{G}$. Будем полагать, что исходные данные подаются на вход алгоритма в следующем виде: $paths = \{(v1_1,v2_1,v3_1,\dots), \{(v1_2,v2_2,v3_2,\dots),\dots\}$ – список путей, vi_j – числовые значения вершины i в пути j, $i = 1..m$, $j = \in\{1,2,\dots,|V|\}$, $E_{subgraph}(v_i,v_j)=\{l|(v_i,v_j)\in G_l\}$ – список принадлежности ребер графа подграфам.

На следующем этапе нужно составить таблицы $M(p_i)$, которые затем будут объединены в таблицу $M(p)$. Для этого необходимо преобразовать каждый из путей вида $(v1_i,v2_i,v3_i,\dots)$ для пути p_i в набор , $(v_{1_j},s_{1_j},v_{2_j},s_{2_j},v_{3_j},\dots v_{l\text{-}1_j},s_{l\text{-}1_j},v_{l_j})$, $s_{kj} \in E_{subgraph}(v_{kj}, v_{k+1_j})$ множество которых $path_{mult}$ содержит все возможные варианты прохождения по подграфам в этом пути. Опишем простой алгоритм для данного действия.

Алгоритм 1. Генерация вариантов прохождения по подграфам в пути

1. Пусть имеется путь $p = (v1,v2,v2,\dots)$ и таблица $E_{subgraph}$, описанная выше. Определим пустую таблицу $path_{mult} = ( )$
2. Помещаем $v1$ в таблицу: $path_{mult} = (v1)$. Если $E_{subgraph}(v1,v2) = \{s1\}$, то помещаем s1 в таблицу: $path_{mult} = (v1,s2)$. Если $E_{subgraph}(v1,s2) = \{s1_1,s1_2,\dots,s1_m\}$, тогда составляем таблицу следующего вида:

$$\begin{pmatrix} v1 & s1_1 \\ v1 & s1_2 \\ \dots & \dots \\ v1 & s1_m \end{pmatrix}$$

3. Помещаем *v*2 в таблицу путем добавления ее справа к каждой строке таблицы. Продолжаем добавлять вершины и числа принадлежности ребра к подграфу до тех пор, пока не встретится $|E_{subgraph}(v_i, v_{i+1})| = |\{si_1, si_2, \ldots, si_z\}| > 1$
4. Пусть на *i*-м шаге имеется таблица следующего вида:

$$\begin{pmatrix} v_1 & s_{1_1} & v_2 & s_2 & v_3 & \ldots & v_i \\ v_1 & s_{1_2} & v_2 & s_2 & v_3 & \ldots & v_i \\ & & \ldots & & & & \\ v_1 & s_{1_m} & v_2 & s_2 & v_3 & \ldots & v_i \end{pmatrix}$$

Тогда добавляем значения $\{si_1, si_2, \ldots, si_z\}$ следующим образом:
Далее переходим к шагу 2, с вершиной v_{i+1} вместо v_1.

$$\begin{pmatrix} v_1 & s_{1_1} & v_2 & s_2 & v_3 & \ldots & v_i & s_{i_1} \\ v_1 & s_{1_2} & v_2 & s_2 & v_3 & \ldots & v_i & s_{i_1} \\ & & \ldots & & & & & \\ v_1 & s_{1_m} & v_2 & s_2 & v_3 & \ldots & v_i & s_{i_1} \\ v_1 & s_{1_1} & v_2 & s_2 & v_3 & \ldots & v_i & s_{i_2} \\ v_1 & s_{1_2} & v_2 & s_2 & v_3 & \ldots & v_i & s_{i_2} \\ & & \ldots & & & & & \\ v_1 & s_{1_m} & v_2 & s_2 & v_3 & \ldots & v_i & s_{i_{z-1}} \\ v_1 & s_{1_1} & v_2 & s_2 & v_3 & \ldots & v_i & s_{i_z} \\ v_1 & s_{1_2} & v_2 & s_2 & v_3 & \ldots & v_i & s_{i_z} \\ & & \ldots & & & & & \\ v_1 & s_{1_m} & v_2 & s_2 & v_3 & \ldots & v_i & s_{i_z} \end{pmatrix}$$

5. Алгоритм продолжается до тех пор, пока не закончатся вершины в пути.

Далее работа сводится к простым операциям конкатенации таблиц и заполнения их, следуя описанным в оригинальной статье принципам.

ДЕКОМПОЗИЦИЯ

Перейдем непосредственно к алгоритму декомпозиции. Здесь и далее будут использоваться обозначения из [1]. Как было упомянуто выше, здесь есть два момента, связанных с эвристиками локального поиска.

Во-первых, это сортировка по N_Σ. Чтобы ускорить обработку данных, сортировку по N_Σ можно не проводить, если выполняются следующие условия:

1. после сортировки по столбцам mk, c и bal_Σ верхняя строка таблицы M, где в столбце mk находится «1», единственная;
2. после сортировки по столбцам mk, c и bal_Σ верхние строки таблицы M, где в столбце mk находится «1» для одного и того же p содержат хотя бы два различных значения в столбце c;
3. после сортировки по столбцам mk, c и bal_Σ верхние строки таблицы M, где в столбце mk находится «1» для одного и того же p содержат хотя бы два разных значения в столбце bal_Σ.

Если ни одно из вышеперечисленных условий не выполняется, расчет N_Σ организован следующим образом:

1. для каждой строки из множества $M(p_i)$ добавляем ее во множество $M(D)$ – множество отобранных в декомпозицию строк на *i*-м шаге;
2. рассчитываем N_Σ;
3. для невыбранных на *i*-м этапе строк значения N_Σ можно не рассчитывать, так как после сортировки они всё равно окажутся «ниже», чем значения из $M(p_i)$. Для определенности можно считать, что они равны ∞.

Второй эвристический алгоритм – это расчет баланса. Перепишем расчет баланса в вид алгоритма, который можно будет легко перевести в программный код:

Алгоритм 2. Расчет баланса

1. Пусть имеется таблица M после выбора строки в декомпозицию, nom – строка таблицы, выбранная на предыдущем шаге, $p(nom)=p_i$, $Ind(nom)=Ind_i$.
2. Для каждого s из $[1..K]$ заведем массив для условия отбора строк $Cond(s)$ с размерностью $|M|\times 1$, где $|M|$ – количество строк в таблице M.
3. Для каждого s из $[1..K]$ и для каждого i из $[1..l-1]$ посчитаем условие следующего вида $cond_i$ (z) = Истина, если в *z*-й строке таблицы *M*: *vi* = *vi*(*nom*) и *vi*+1= *vi*+1(*nom*) и *si* = *s*. Далее, выполняется построчное ИЛИ для строк из $cond_i$ и $Cond$.
4. Для каждого *s* из $[1..K]$ отбираем строки из таблицы *M* по условию: $M_z[p] \neq pi$ И $M_z[mk] = 1$ И $M_z[bal_s] = 0$ И $M_z[Ind] \neq Ind_i$ И $Cond(s)$ и выставляем значение в столбце bal_s равным 1 для отобранных строк.

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

В качестве языка программирования был выбран Python 3. Этот язык обладает простым синтаксисом, удобными для работы типами данных, а также для него были реализованы математические пакеты с хорошим API (numpy) и мощное средство для работы с таблицами (pandas). Функции из этих пакетов лягут в основу программного кода для реализации алгоритма декомпозиции путей ориентированного графа. Кроме этого, будет использована библиотека graphviz для визуализации графа. При реализации будет использован объектно-ориентированный подход. Он позволит объединить в одном месте данные и код для работы с ними, а также скрыть детали реализации алгоритма от пользователей (принцип инкапсуляции). Входные данные будут подаваться через конструктор, в котором будет происходить создание исходной таблицы, а единственным, кроме конструктора, открытым методом будет метод, в котором происходит непосредственно декомпозиция.

Для тестирования практической реализации алгоритма декомпозиции необходимо сгенерировать сильно связный ориентированный граф и множество путей.

Пусть имеется граф $\vec{G}$, изображенный на рис. 1 и 2.

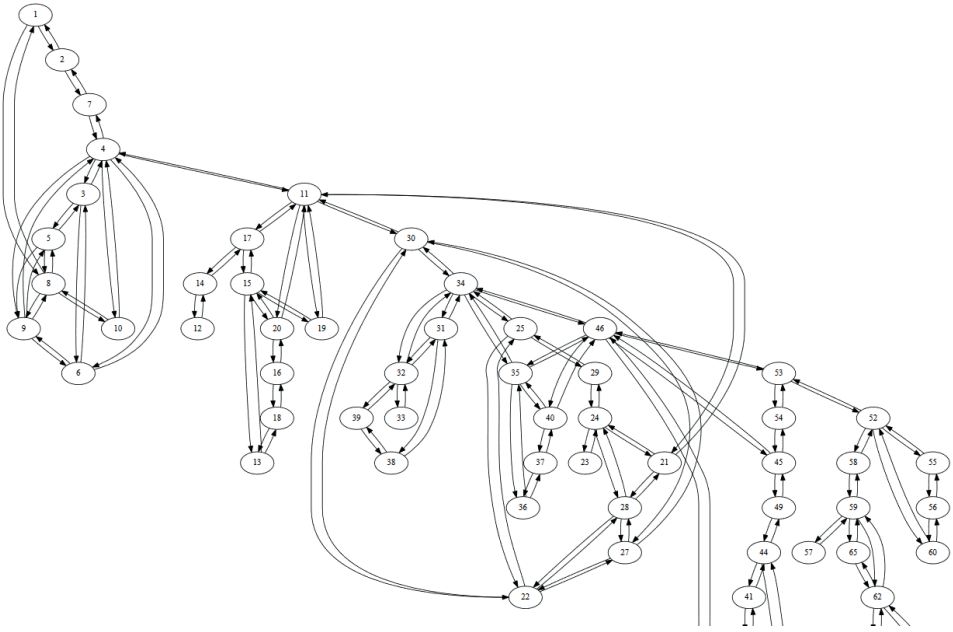

Рис. 1. Сильно связный граф (1)

Выберем множество путей

$$P = \{$$

$$p_1 = \{1,2,7,4,11,30,22,25,34,46,53,52,58,59,65,62,70,67\},$$

$$p_2 = \{11,17,15,13,18,16,20,11,30,27,22,30,34,46\},$$

$$p_3 = \{38,31,34,30,11,21,28,22,30\},$$

$$p_4 = \{1,2,7,4,3,5,8,9,6\},$$

$$p_5 = \{5,3,4,7,2,1\}$$

$$\}$$

Будем считать, что граф состоит из 10 подграфов, $G_1,\ldots G_{10}$

$$V(G_1) = \{1,2,3,4,5,6,7,8,9,10,11,30)$$

$$V(G_2) = \{11,12,13,14,15,16,17,18,19,20,30\}$$

$$V(G_3) = \{22,30,31,32,33,34,38,39,46\}$$

$$V(G_4) = \{11,21,22,23,24,25,27,28,29,30,34,35,36,37,40,46\}$$

$$V(G_5) = \{41,42,43,44,45,46,47,48,49,50,53,54\}$$

$$V(G_6) = \{52,53,54,55,56,57,58,59,60,62,65\}$$

$$V(G_7) = \{61,62,63,64,65,66,67,68,69,70,71,75\}$$

$$V(G_8) = \{66,68,71,72,73,74,75,76,77,78,79,80,85\}$$

$$V(G_9) = \{81,82,83,84,85,86,87,88,89,90\}$$

$$V(G_{10}) = \{85,88,91,93,94,95,96,97,96,97,98,99,100\}$$

$$E(G_l) = \{(v_i, v_j) | v_i, v_j \in G_l\},\ l \in \{1,\ldots,10\}$$

Для $G_1,\ldots G_{10}$, $E\left(G_l\right)=\{\left(v_i,v_j\right)|v_i,v_j\in G_l\}$, $l\in\{1,\ldots,10\}$

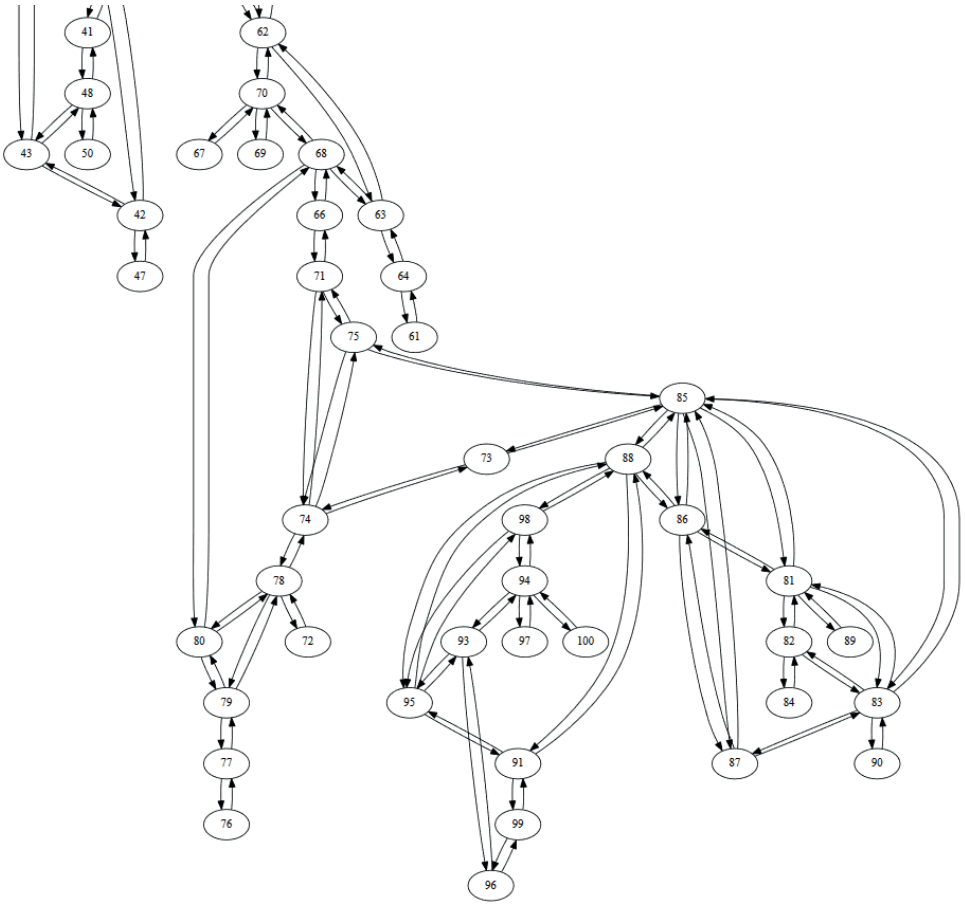

Рис. 2. Сильно связный граф (2)

Пути из P лежат в $G_1,\ldots G_7$.

Составим таблицу $E_{subgraph}$, которая используется в алгоритме 1. После выполнения алгоритмов 1 и 2 и алгоритма, описанного в [1] получаем, что пути находятся в следующих подграфах:

$$p_1\in\{G_1\},p_2\in\{G_3,G_4\},p_3\in\{G_2,G_3\},p_4\in\{G_3,G_5,G_6,G_7\},p_5\in\{G_1\}$$

Работа алгоритма для графа выше занимает в среднем 6 секунд.

Результаты запусков алгоритма для разных входных данных приведены в табл. 1.

Таблица 1

Результаты запусков алгоритма декомпозиции

Количество вершин	Количество ребер	Количество путей	Количество подграфов	Время работы алгоритма
6	5	1	1	20 мс
6	14	4	3	800 мс
16	28	2	3	1.2 с
27	36	2	7	1.6 с
70	130	5	7	6 с
100	160	7	10	11 с

ЗАКЛЮЧЕНИЕ

Разработаны алгоритмы для подготовки таблицы для декомпозиции и подробнее описаны алгоритмы декомпозиции. Полученные результаты реализованы в виде программного кода и проверены для ориентированного графа размерности 100 (вершин).

Литература

1. *Гайнанов Д.Н., Коныгин А.В., Рассказова В.А.* Моделирование грузовых железнодорожных перевозок методами теории графов и комбинаторной оптимизации // Автоматика и телемеханика. 2016. № 11. С. 60–79.
2. *Гайнанов Д.Н., Кибзун А.И., Рассказова В.А.* Теоретико-графовый алгоритм решения задачи о назначении и перемещении локомотивов // Вестник компьютерных и информационных технологий. 2017. № 5. С. 51–56.
3. *Гайнанов Д.Н., Кибзун А.И., Рассказова В.А.* Задача о декомпозиции множества путей ориентированного графа и ее приложение // Автоматика и телемеханика. 2018. № 12. С. 142–166.

Practical Realization of Algorithm of Oriented Graph Paths Decomposition

Igor A. Zolotarev*
Moscow Aviation Institute, Moscow, Russia,
ORCID: https://orcid.org/0000–0002–6437–2212
e-mail: yngvar.antonsson@gmail.com

Varvara A. Rasskazova**
Moscow Aviation Institute, Moscow, Russia,
ORCID: https://orcid.org/0000–0003–4943–3133
e-mail: varvara.rasskazova@mail.ru

This study aims to clarify the methodological status program realization of oriented graph paths decomposition. Algorithms of matrix formulation of decomposition table M from multitude of paths, table sorting by N_{Σ} and balance computing explored. On the basis of those algorithms and original algorithm of oriented graph paths decomposition realized Python 3 program. Results for random-generated graph size of 100 vertices computed and time measured. The results obtained can be used to solve the problem of organizing freight rail transportation at the stage of assignment and movement of locomotives. The scientific and practical novelty of the work lies in a significant reduction in the dimension of the original problem, which is especially important in the conditions of transport networks of complex topology.

Keywords: optimization theory, graph optimization, algorithm of directed graph paths decomposition, strongly connected graph, subgraphs.

For citation:
Zolotarev I.A., Rasskazova V.A. Practical Realization of Algorithm of Oriented Graph Paths Decomposition. *Modelirovanie i analiz dannykh = Modelling and Data Analysis*, 2020. Vol. 10, no. 3, pp. 60–68. DOI: https://doi.org/10.17759/mda.2020100305 (In Russ., abstr. in Engl.).

References

1. Gainanov D.N., Konygin A.V., Rasskazova V.A. Modelling railway freight traffic using the methods of graph theory and combinatorial optimization. *Automation and Remote Control*, 2016 vol. 77, no. 11, pp. 1928–1943.
2. Gainanov D.N., Kibzun A.I., Rasskazova V.A. Theoretical-graph algorithm in the problem on the assignments and transportations of locomotives. *Herald of computer and information technologies*, 2017, no. 5, pp. 51–56.

***Igor A. Zolotarev,** Master Student, Moscow Aviation Institute, Moscow, Russia, ORCID: https://orcid.org/0000–0002–6437–2212, e-mail: yngvar.antonsson@gmail.com

****Varvara A. Rasskazova,** PhD in Physics and Mathematics, Associate Professor, Moscow Aviation Institute, Moscow, Russia, ORCID: https://orcid.org/0000–0003–4943–3133, e-mail: varvara.rasskazova@mail.ru

3. Gainanov D.N., Kibzun A.I., Rasskazova V.A. The Decomposition Problem for the Set of Paths in a Directed Graph and Its Application. *Automation and Remote Control*, 2018, vol. 79, no. 12, pp. 2217–2236.