ОПРЕДЕЛЯНЕ НА СЛОЖНОСТ НА ИТЕРАЦИОНЕН АЛГОРИТЪМ ЗА ДЕКОМПОЗИЦИЯ ГРАФИ

Матьо Динев[1], Валентина Кукенска[1], Петър Минев[1], Илиян Върбов[1]

[1]Технически Университет - Габрово

DETERMINING THE COMPLEXITY OF AN ITERATIVE GRAPH DECOMPOSITION ALGORITHM

Matyo Dinev[1], Valentina Kukenska[1], Petar Minev[1], Ilian Varbov[1]

[1] Technical University of Gabrovo

Abstract

This paper presents a methodology for determining the complexity assessment of an iterative graph decomposition algorithm. For this purpose, defined rules were used, which were applied to the procedural steps of the algorithm. By applying BIG-O notation, to the assessment, the type of time complexity is also obtained.

The methodology for determining the complexity assessment is applied to graphs of different sizes. Based on this, the maximum number of operations for the algorithm was calculated. Based on the obtained results, conclusions about the iteration algorithm are drawn. Based on this, the maximum number of operations for the algorithm was calculated. Based on the obtained results, conclusions about the iteration algorithm are drawn.

Keywords: iterative algorithm; decomposition, graph; algorithm complexity.

ВЪВЕДЕНИЕ

Декомпозицията е оптимизационна задача за разделянето на граф на подграфи. За решаването ѝ се използват следните критерии: минимален брой външни връзки между подграфите; минимален брой подграфи; определен брой елементи в подграфите [1,2,3].

Математическата формулировка на задачата за декомпозиция на граф е следната:

Даден е графът $G(V,E)$, където $V = (V_1, V_2, V_3, \ldots, V_n)$ е множество на върховете, а $E = (E_1, E_2, E_3, \ldots, E_m)$ множество на ребрата. Да се раздели графа на N подграфи, където $G_p=(V_p, E_p)$, $p=1,2,3,\ldots,N$ е множество на отделените подграфи. Търси се такова разделяне на графа G на подграфи, при което се удовлетворяват на следните условия:

- Обединяването на всички подграфи да води до първоначалния граф G;

$$\cup G_p = G$$

- Сечението на множеството на върховете на отделните подграфи да води до празно множество;

$$V_i \cap V_j = \emptyset$$

При разделянето на графа на подграфи, ребрата свързващи върховете в даден подграф се приемат за вътрешни, а тези между отделните подграфи – за външни. Реброто E_i е вътрешно, ако $E_i \in E_p$, в противен случай се счита за външно [2,3].

За декомпозирането на графи се използват последователни, итерационни и смесени алгоритми. За всеки от тях може да се определи оценка на неговата сложност. Сложност на алгоритъм представлява изменението на необходимото процесорно време или памет при изпълнение на алгоритъма, докато се промяна размера на входните данни. В настоящия доклад е обърнато внимание на сложност по време [3,4,5,6,7,8].

Бързодействието на алгоритъма зависи от два параметъра. Първият е броят на операциите които процесора изпълнява за 1 секунда. Вторият параметър е необходимият

брой операции за изпълнението на алгоритъма. Определящият критерий е именно вторият.

Броят на необходимите операции може да се определи по един от следните случаи [3,5,6,7]:

- Най – лош случай (worst-case);
- Среден случай (average-case);
- Най – добър случай (best-case);

Най-лошият случай дава информация за максималният брой операции необходими за изпълнението на алгоритъма, а най-добрият – за минималния брой операции [3,5,6]. Целта при определянето оценката на сложност на алгоритмите е да се получи информация за максималния брой операции. Поради тази причина най-лошият случай е най-често използвания.

Съществуват няколко асимптотични нотации за сложност на алгоритмите като: Big-O нотация, Big-θ (big theta) нотация, Big-Ω (big omega) нотация. В настоящия доклад е разгледана само Big-O нотацията. При нея съществува горна граница за времето на изпълнение на един алгоритъм, при достатъчно големи входни данни. Това означава, че Big-O нотацията се прилага с worst-case и се получава информация за максималния брой операции в изследвания алгоритъм [3,5,6].

При определянето на сложността на алгоритъм се вземат под внимание следните правила.

За една операция се приемат [3,5,7,8,9]:

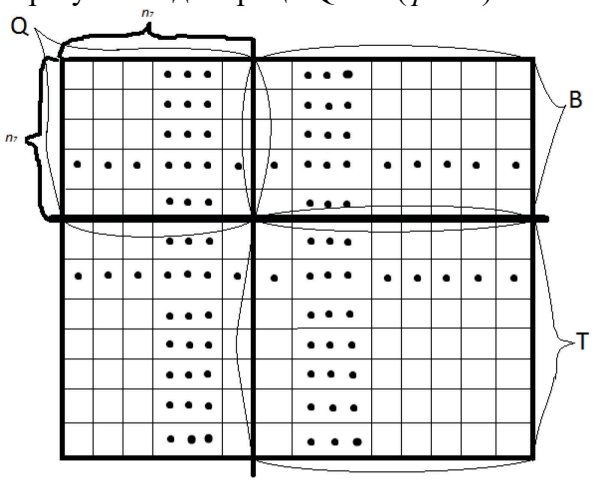
- Дефиниране и присвояване на стойност на променлива;
- Сравняване;
- Аритметична операция;
- Добавяне или вземане на елемент от масив (множество, граф и т.н.);
- Извикване на метод или функция.

В настоящия доклад е определена оценката на сложност на съществуващ итерационен алгоритъм за декомпозиция на графи. За целта са използвани дефинираните правила, които са приложени върху процедурните стъпки на алгоритъма. Чрез прилагане на BIG-O нотация, върху оценката, се получава и вида на времевата сложност

Получената оценка на сложност е приложена върху графови модели с различни размери. На базата на това е изчислен максималният брой операции за алгоритъма.

ИТЕРАЦИОНЕН АЛГОРИТЪМ ЗА ДЕКОМПОЗИЦИЯ НА ГРАФ

За итерационния алгоритъм се създава матрицата на съседство R за графа. От нея се образуват подматрици Q и T (*фиг.1*).



***Фиг. 1**. Декомпозирана матрица на съседство*

Пресмятат се елементите на вектор α по следната формула: [10,11,12]

$$\begin{cases} \alpha_i = s_i(B) - s_i(Q) \text{ ако } i \in (1, n_1) \\ \alpha_i = s_i(Q!) - s_i(\mathrm{T}) \text{ ако } i \in (n_1, n) \end{cases} \quad (1)$$

където

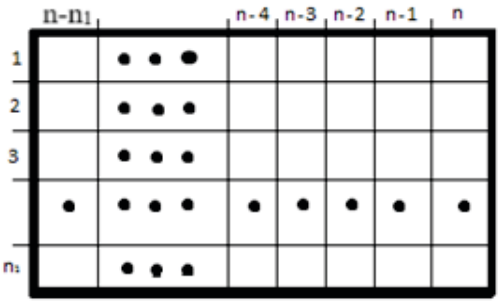
$s_i(Q)$ – сумата на елементите в под-матрицата Q за *i*-тия ред;

$s_i(B)$ – сумата на елементите в под-матрицата B за *i*-тия ред;

$s_i(Q!)$ – сумата на елементите в под-матрицата под главния диагонал за *i*-тия ред

$s_i(T)$ – сумата на елементите в под-матрицата T за *i*-тия ред.

Проверяват се елементите на вектора α дали има положителни стойности. Ако във вектор α се съдържат стойности по – големи или равни на 0, то има възможност за итерационно подобрение. За целта се построява матрица B, чиято структура съответства на показаната на *фиг.2* [10,11,12].



***Фиг. 2**. Матрица B*

Елементите на тази матрица се пресмятат по формула (2).

$$b_{ij} = \alpha_i + \alpha_j - 2 * r_{ij}\,, \qquad (2)$$

Избира се елементът с най – голяма положителна стойност. Определят се номерата на реда и стълба, в които той е разположен. Те определят двойката върхове, които ще бъдат в различни подграфи. Матрицата R се преобразува като се разменят редовете и стълбовете, съответстващи на двойката върхове за размяна [10,11,12].

СТЪПКИ НА АЛГОРИТЪМА

1. Образуване на матрицата на съседство R за графа G и под-матриците Q и T.
2. Определяне елементите на вектора α.
3. Проверка дали всички стойности на вектора α са отрицателни. Ако е вярно, преминаваме към стъпка 8. Ако не преминаваме към стъпка 4.
4. Образува се матрица B и се пресмятат нейните елементи по формула (2).
5. Търси се най – големият елемент от матрицата B. Определящи са реда и стълба, в който този елемент е разположен.
6. Преобразува се матрицата на съседство R.
7. Връщане към стъпка 2.
8. Включване на върховете отговарящи на редовете от подматрицата Q, в множеството G_1. Останалите върхове се включват в множество G_2. Отделните множества съдържат върховете в отделните подграфи.

ОПРЕДЕЛЯНЕ СЛОЖНОСТТА НА ИТЕРАЦИОННИЯ АЛГОРИТЪМ

За да се определи оценката на сложност на итерационния алгоритъм е необходимо да се направи анализ на всяка една негова стъпка, разглеждана в най-лошия ѝ случай. Оценката на съответната стъпка е отбелязана с буквата O и долен индекс, състоящ се от първата буква от името на алгоритъма, и от номера на съответната стъпка.

1. За попълването на всяка клетка от матрицата $\boldsymbol{R}$ се присвоява по една стойност. Присвояването се брои за една операция. За тази стъпка са необходими n^2 брой операции. $\boldsymbol{O_{и1}=n^2}$
2. В стъпка 2 за всеки n елемента на вектора α се правят n на брой събирания или изваждания по формула (1). За тази стъпка са необходими отново n^2 на брой операции. $\boldsymbol{O_{и2}=n^2}$
3. Проверката на всеки от елементите на вектора α се извършва чрез операция за сравнение. За стъпката са необходими n на брой операции. $\boldsymbol{O_{и3}=n}$
4. Пресмятането на елементите на матрицата $\boldsymbol{B}$ се извършва чрез формула (2). За определянето на един елемент от матрицата са необходими три операции: събиране, изваждане и присвояване. За стъпката са необходими $3*k$ на брой операции, където k е броя на елементите на матрица $\boldsymbol{B}$. В най-лошия случай броя на елементите на матрицата ще бъде $(\frac{n}{2})^2$. Максималният брой операции необходими за тази стъпка е $\boldsymbol{O_{и4}=\frac{3n^2}{4}}$
5. Определянето на най-големия елемент от матрицата $\boldsymbol{B}$ се извършва с k на брой сравнявания.

   $$k = (\frac{n}{2})^2 = \frac{n^2}{4}$$

   Определянето на позицията на елемента се извършва чрез извикване на функция. Максималният брой операции се определя със следния израз:

   $$\mathrm{O_{и5}} = \boldsymbol{\frac{n^2}{4} + 1}$$

6. Размяната на местата на два елемента в матрицата R се изразява чрез три операции за присвояване. За размяната на два реда и два стълба са необходими съответно $3*n$ на брой операции. Максималният брой операции за тази стъпка се изразява чрез следния израз:

   $$\mathrm{O_{и6}} = \boldsymbol{3n + 3n = 6n}$$

7. В тази стъпка се прави връщане към стъпка 2. $\boldsymbol{O_{и7}=O_{и2}=n^2}$.
8. Добавянето на елемент към множество се приема за една операция. Броят на върховете на графа е n и те ще се разпределят в отделни множества. За тази стъпка максималният брой операции е $\boldsymbol{O_{и8}=n}$.

Максималният брой операции на итерационния алгоритъм се получава като сума от оценките на всяка стъпка.

$O_n=O_{u1}+O_{u2}+O_{u3}+O_{u4}+O_{u5}+O_{u6}+O_{u7}+O_{u8}$ *(3)*

След преобразуване на формула (3) се получава:

$$O_и = 4n^2 + 8n + 1 \qquad (4)$$

След прилагането на Big-O нотацията върху формула (4) се вижда, че сложността на алгоритъма е квадратична ***O(n²)***.

РЕЗУЛТАТИ И ИЗВОДИ

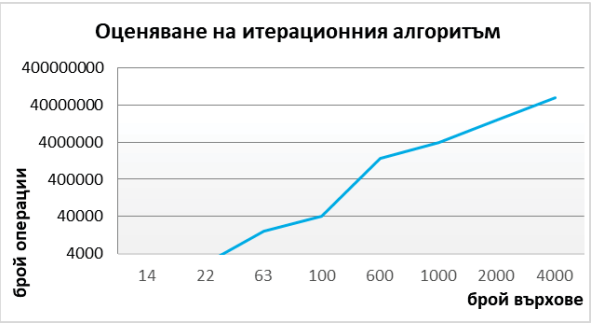
С предложената методика е определена оценката за сложност на итерационен алгоритъм за декомпозиция. Изчислени са максималният брой операции на алгоритъма при декомпозиция на графи с различни размери (брой върхове от 14 до 4000). Получените резултати са представени в *табл.1*.

Табл.1.*Максимален брой операции на алгоритъма*

N	***$O_и$***
14	897
22	2 113
63	16 381
100	40 801
600	1 444 801
1 000	4 008 001
2 000	16 016 001
4 000	64 032 001

ЛЕГЕНДА
n *– брой върхове в графа;*
$O_и$ *– максимален брой операции за итерационния алгоритъм;*

На *фиг.3* резултатите са представени графично.



Фиг.3 *Графично представяне на максималния брой операции на алгоритмите*

Въз основа на горните резултати може да се направи извода, че с увеличаване броя на върховете в графа многократно нараства и максималния брой операции необходими за работата на алгоритъма. Това означава, че времето му за изпълнение, също ще се увеличава с увеличаване броя на върховете.

ЗАКЛЮЧЕНИЕ

Предложена е методика за оценка на сложността на итерационния алгоритъм за декомпозиция на графи. Чрез нея е определен максималният брой операции на алгоритъма. Получените резултати могат да се използват за сравнение с други алгоритми за декомпозиция.

Итерационният алгоритъм за декомпозиция може да се приложи при решаване на задачата за компоновка при конструктивно проектиране на схеми, устройства и системи. На базата на аналогия може да се построи топологичен граф, върховете на които съответстват на елементите на схемата, а ребрата на връзките между тях. При решаване на тази задача като критерии се използват минимален брой връзки между конструктивните модули, максимална степен на свързаност между елементите за всяка отделна част и други.

БЛАГОДАРНОСТИ

Настоящият документ е изготвен с финансовата помощ на договор № 2209Е за провеждане на научни изследвания по проект на тема: „Виртуална лаборатория за обучение по проектиране на цифров хардуер“ към Технически университет – Габрово.

REFERENCE

[1]. Andonian M.O., Decomposition of graphs and application in structural analysis of complex systems, Author abstract of a dissertation at the Higher Mechanical and Electrical Engineering Institute "V.I. Lenin", Sofia, 1976
[2]. Kukenska V., Development of a modular teaching-research dialog system for automated circuit design in electronics., Dissertation, Gabrovo, 1998
[3]. Dinev M., Research of decomposition algorithms., Dissertation, Gabrovo Bulgaria, May 2023
[4]. Soltys M., An Introduction to the Analysis of Algorithms 2nd Edition, New Jersey: World Scientific, 2012.
[5]. Nakov P., P. Dobrikov, Programming = ++Algorithms, TopTeam Co., Sofia, 2015.

[6]. Stoichev S. D., Synthesis and analysis of algorithms with Pascal and C++ programs, BPS, Sofia, 2008.
[7]. Dinev M., V. Kukenska, P. Minev, Complexity of decomposition algorithm, International Scientific Conference Unitech'2018 – Gabrovo, 16 – 17 November 2018, Volume II, pp. 151-155, ISSN 1313-230X.
[8]. Dinev M., Metodology for evaluation of decomposition algorithms, International Scientific Conference Unitech'2020 – Gabrovo, 20 – 21 November 2020, Volume I, pp. 384-389, ISSN 1313-230X.
[9]. Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest., Introduction to Algorithms. Cambridge, MA: MIT Press, 1990.
[10]. Kukenska V., P. Minev, Automation of engineering work. Manual for laboratory exercises, "Vasil Aprilov" University Publishing House, Gabrovo, 2008.
[11]. Kukenska V. P. Minev Automation of engineering work, University Publishing House "Vasil Aprilov" Gabrovo 2016.
[12]. Dinev M.,V. Kukenska, Matlab application for decomposition of graphs, XII International Conference Strategy of Quality in Industry and Education, Varna, Bulgaria, May 30 – June 02 2016, pp.537-542, ISBN 978-966-2752-71-7.