Известия Саратовского университета. Новая серия. Серия: Математика. Механика. Информатика. 2021. Т. 21, вып. 3. С. 400–407
Izvestiya of Saratov University. Mathematics. Mechanics. Informatics, 2021, vol. 21, iss. 3, pp. 400–407
https://mmi.sgu.ru https://doi.org/10.18500/1816-9791-2021-21-3-400-407
Article

The search for minimal edge 1-extension of an undirected colored graph

P. V. Razumovsky

Saratov State University, 83 Astrakhanskaya St., Saratov 410012, Russia

Peter V. Razumovsky, shprotby@gmail.com, https://orcid.org/0000-0003-3648-3166

Abstract. Let $G = (V, \alpha, f)$ be a colored graph with a coloring function f defined on its vertices set V. Colored graph G^* is an edge 1-extension of a colored graph G if G could be included into each subgraph taking into consideration the colors. These subgraphs could be built from G^* by removing one of the graph's edges. Let colored edge 1-extension G^* be minimal if G^* has as many vertices as the original graph G and it has the minimal number of edges among all edge 1-extensions of graph G. The article considers the problem of search for minimal edge 1-extensions of a colored graph with isomorphism rejection technique. The search algorithm of all non-isomorphic minimal edge 1-extensions of a defined colored graph is suggested.

Keywords: graph extensions, graph colorings, colored graphs, edge extensions, minimal extensions, graph isomorphism, colored graph isomorphism, fault tolerance

For citation: Razumovsky P. V. The search for minimal edge 1-extension of an undirected colored graph. *Izvestiya of Saratov University. Mathematics. Mechanics. Informatics*, 2021, vol. 21, iss. 3, pp. 400–407 (in English). https://doi.org/10.18500/1816-9791-2021-21-3-400-407

Научная статья
УДК 517.98

О поиске минимальных реберных 1-расширений неориентированного цветного графа

П. В. Разумовский

Саратовский национальный исследовательский государственный университет имени Н. Г. Чернышевского, Россия, 410012, г. Саратов, ул. Астраханская, д. 83

Разумовский Петр Владимирович, аспирант кафедры теоретических основ компьютерной безопасности и криптографии, shprotby@gmail.com, https://orcid.org/0000-0003-3648-3166

Аннотация. Граф $G = (V, \alpha, f)$ — это цветной граф с определенной на множестве его вершин функцией раскраски f. Цветной граф G^* называется реберным 1-расширением цветного графа G, если граф G можно вложить с учетом цветов в каждый граф, получающийся из графа G^* удалением любого его ребра. Реберное 1-расширение G^* графа G называется минимальным, если граф G^* имеет столько же вершин, сколько содержит исходный граф

G, а среди всех реберных 1-расширений графа G граф G^* имеет минимальное число ребер. Рассматривается задача поиска минимальных реберных 1-расширений цветного графа без проверки на изоморфизм. Предлагается алгоритм поиска множества всех неизоморфных минимальных 1-расширений для заданного цветного графа.

Ключевые слова: расширения графов, раскраски графов, цветные графы, реберные расширения, минимальные расширения, изоморфизм графов, изоморфизм цветных графов, отказоустойчивость

Для цитирования: *Razumovsky P. V.* The search for minimal edge 1-extension of an undirected colored graph [*Разумовский П. В.* О поиске минимальных реберных 1-расширений неориентированного цветного графа] // Известия Саратовского университета. Новая серия. Серия: Математика. Механика. Информатика. 2021. Т. 21, вып. 3. С. 400–407. https://doi.org/10.18500/1816-9791-2021-21-3-400-407

Introduction

There are many vital areas where high-technological computing devices and systems are used. The importance of these systems creates high availability requirements for them. These requirements could be met if system architecture is designed, which should satisfy all demands. One of these demands is fault tolerance. Fault tolerance term was proposed by A. Avizienis [1] and defines the property that enables a system to continue operating properly in the case of some components failure.

Fault tolerance research was made by J. P. Hayes [2] and had a huge impact. Hayes researched critical faults when failed components were totally removed from the system, and suggested considering fault tolerance problems on a graph model. Such a model considers critical fault as a vertex removal from a graph. Moreover, Hayes proposed several algorithms for building k-fault tolerance implementation of the system. k-fault means that system continues operating properly in the event of the failure of k components at one time.

In the 90s, the famous American mathematician Frank Harary together with John P. Hayes [3, 4] generalized a fault tolerance graph model in the case of connections faults between system components. They proposed this model as an edge fault tolerance system implementation.

Let the system be an edge k-fault tolerance implementation of an original system if the fault of any k edges leads to the graph which includes the graph of an original system. Edge k-fault tolerance system implementation could be described as adding extension connections between components. Extension items are masked in regular system work. When a fault occurs, the system reconfigures itself to achieve an original state. M. B. Abrosimov [5] suggested calling the edge k-fault tolerance system implementation an edge k-extension of a graph.

There is proof that finding edge k-extension of a graph is the NP-complete problem [6]. At the beginning of fault tolerance research [5], a backtracking algorithm of generating all minimal edge k-extensions of graphs was suggested which will be referred to as ME-kE. The algorithm uses brute force of all possible solutions and is hard in multithread implementation, but it is universal and convenient for modifying.

One of the most effective improvements of the backtracking algorithm is an isomorphism rejection technique [7]. This technique allows us to reject all isomorphic graphs

and keeps only non-isomorphic. Moreover, it allows us to get rid of comparing each of the two graphs on isomorphism.

There are two effective and efficient approaches to this technique. The first approach was described in the Read – Faradzhev method [7], the second one was proposed by McKay [6].

M. B. Abrosimov, H. H. K. Sudani and A. A. Lobov [8] suggested the algorithm of construction of all minimal edge extensions of the graph with isomorphism rejection. I. A. K. Kamil, M. B. Abrosimov and A. A. Lobov [9] proposed the similar results for minimal vertex extensions.

1. The search problem for edge extensions of colored grpahs

This article is a logical continuation of edge extensions research and considers the problem of search for edge extensions of graphs with defined coloring function. There are various ways to define the problem for colored graph edge extensions. We can choose different coloring functions for graphs: vertex coloring, edge coloring, combined, i.e. vertices with edges coloring. This article considers only edge extensions for graphs with vertex coloring. Let us define several common terms. We are considering only undirected graphs.

1.1. Colored graphs and edge extensions definitions

Definition 1. Let $G = (V, \alpha)$ be a graph, and $i \in \mathbb{N}$. Then function $f : V \to \{1, \ldots, i\}$ is vertex i-coloring of graph G and $f(v), v \in V$ is the color of vertex v. Graph with coloring function is usually referred to as a graph with colored vertices, or a *colored graph*. Colored graphs notation looks like $G = (V, \alpha, f)$.

Definition 2. Graph $G_R = (V_R, \alpha_R, f_R)$ is *edge k-extension* (E-kE) implementation of the i-colored graph $G = (V, \alpha, f)$, where $k \in \mathbb{N}$ if graph G could be embedded with the preservation of its colors into each subgraph of G_R which could be constructed by removing any of its k edges.

In other words, graph G should be embedded in any subgraph of G_R constructed by removing its any k edges, with preservation of the original coloring function.

Definition 3. Graph $G^* = (V^*, \alpha^*, f^*)$ is *minimal edge k-extension* (ME-kE) implementation of the i-colored graph $G = (V, \alpha, f)$, where $k \in \mathbb{N}$ if the following criteria are satisfied:

1) graph G^* is an edge k-extension of the colored graph G;

2) graph G^* has the same number of vertices as the original graph G;

3) the number of edges in G^* set is the minimal around all graphs, which met the first two criteria.

In the case of specified coloring function, edge-fault tolerant implementation of the graph should have a special property. This property is the preservation of the coloring of an original graph after reconfiguration. Such property can be formulated in the context of some computing system with different types of components. The edge-fault tolerant implementation of the original system must preserve the number of the components of different types and connections between them.

The problem of preserving original coloring is a non-deterministic computational complexity so it has no efficient solution. This article suggests some ideas to accelerate the search for extensions search with a coloring preservation property.

1.2. Isomorphism rejection approach for the search of edge extensions

Despite [8, 9], we are considering different methods to achieve a solution for an edge extensions search problem. This paper uses the Brendan McKay method [6] of an isomorphism rejection. The main feature of the method is a specific way of candidates generation. A candidate is some structure in this context, e.g. graph, system, device, etc. Generation happens iteratively from low to high: from an ancestor to a descendant. Each candidate could be a descendant of a few ancestors. To implement generation of all structures without isomorphism verification, McKay suggests following these steps.

1. Define the unique canonical ancestor and its own way of generation for each structure.
2. Accept the structure only in the case it is generated by the chosen way from the canonical ancestor.
3. Generate all non-isomorphic structures from the ancestor only once.

Usually, the canonical ancestor form is called the *canonical code* of a structure. Within this article, it is a canonical code of the graph.

Let us define a canonical code and a canonical way to generate structure from. The canonical code is a pair $(\mu_{\max}(G), FC(G))$, where $\mu_{\max}(G)$ is a maximal matrix code of the graph G and $FC(G)$ is a special prime coloring code of the graph G, which we consider below.

2. Maximal matrix code of defined graph as a part of the canonical code

We need to construct maximal matrix code $\mu_{\max}(G)$ for the defined graph as a part of canonical code for the McKay method. The maximal matrix code is a convenient way to define non-isomorphic graphs because it is a graph's complete invariant [10]. As we are considering only undirected graphs, $\mu_{\max}(G)$ will be constructed only by the right upper triangle of an adjacency matrix. Any convenient method to receive maximal matrix code could be chosen.

The applicable way of generation for $\mu_{\max}(G)$ needs to be found. Obviously, it should guarantee that adding another edge between two representative items of different similarity classes does not violate the maximal property of the code. In other words, obtained maximal matrix codes should preserve canonical property. Let us consider vertices similarity definitions in terms of this article [6].

Definition 4. *A colored graph's automorphism* is an isomorphism of a colored graph on itself. In other words, it is a permutation of colored vertices in such a way that the edge set is not changed.

Definition 5. Two vertices are *similar* if and only if there is an automorphism which reflects one vertex to another.

Definition 6. The similar vertices set are called an *orbit*.

The article proposes the algorithm of constructing all new non-isomorphic graphs from a defined maximal matrix code by adding one additional edge. This algorithm meets maximal property requirement.

Let us define function $Orb(G)[v]$ which results in a set of vertices similar to v.

The algorithm's input is $\mu_{\max}(G')$ of graph G' and $f(G')$ is the graph's coloring. The output of the algorithm is a set of maximal matrix codes which are constructed from the original graph by adding one additional edge. The algorithm follows these steps.

1. Calculate an orbit set $Orb(G')$ of the defined graph according to coloring function if it exists.
2. Calculate a set of representative vertices P_{orb} from orbits set, where each item will be the first item from each orbit in $Orb(G')$.
3. For each vertex v define a set Alw_v. It contains vertices in which additional edges are allowed and are based on the following criteria:
   - each vertex from the same orbit as v vertex is allowed;
   - any vertex from the other orbit than v vertex is allowed if it is in P_{orb}, i. e. $p \in Alw_v \Leftrightarrow p \in P_{orb}$, and its number is greater than the number of v vertices.
4. For each vertex v from P_{orb}, we build new codes by adding one additional edge to $\mu_{\max}(G')$ with the following criteria:
   - consider $u \in Alw_v$;
   - $v < u$, where the number of vertex u should be greater than the number of vertex v;
   - $(v, u) \notin \alpha'$.
5. Type all generated codes as maximal matrix codes of non-isomorphic graphs which are constructed from G' by adding one additional edge.

We discover that during the implementation it is more efficient to consider a number of vertices from high to low on step 4.

The described algorithm allows us to specify the canonical way to obtain $\mu_{\max}(G)$ descendants without coloring preservation.

3. Prime coloring code for preserving coloring property in edge extensions

Now let us extend the results above with the coloring preservation requirement's solution. As it was described above, the *coloring code* will be considered. The colors will be coded as integers with zero-indexing. It will help in further implementations.

For an i-colored graph, the *coloring vertex code* of the vertex v is a structure $f(v)\left(|f_0|, |f_1|, \ldots, |f_{i-1}|\right)$, where $f(v)$ is a color of v and $|f_j|$ is a number of j-colored neighbors of vertex v. For example, for some vertex v of the 2-colored graph the coloring vertex code equals to $0(3, 0)$, which means that $f(v) = 0$ and v has three neighbors of color 0 and zero neighbors of color 1.

The vector of such coloring vertex codes structures the *coloring code* of the graph. This coloring code is not convenient because comparing two coloring codes means using some backtracking method which is slow. This demands using a more efficient modification of the coloring code.

Let us choose for each color two unique prime numbers. The first number $fs(f_j)$ could be small, the second number $sc(f_j)$ should be greater than $fs(f_j)^n$, where n is a number of vertices in the graph. Then for each coloring vertex code, we can calculate the number called *prime coloring vertex code*. For the coloring vertex code from the previous example, prime coloring vertex code will be the following:

$$fc(v) = sc(f(v)) \cdot fs(f_0)^{|f_0|} \cdot fs(f_1)^{|f_1|} \cdot \ldots \cdot fs(f_{i-1})^{|f_{i-1}|}.$$

Obviously, this code is a multiplication of prime numbers. Vector of prime coloring vertex codes forms the *prime coloring code* of the defined graph. Let us define it as $FC(G) = \{fc(v) | G = (V, \alpha), v \in V\}$. Therefore graphs' colorings G and G' inclusion check of the preservation of colors could be described in the following algorithm.

Define the function $FC(G)[v] = fc(v)$, where v is a vertex from graph G.

The input of the algorithm are $FC(G)$ non-increasing ordered and $FC(G')$ non-increasing ordered. The result is "YES" or "NO" depending on the answer, whether G coloring is preserved in graph G' or not. The algorithm follows the steps.

1. $i = 0, j = 0$.
2. If $i > |FC(G')|$ and $j > |FC(G)|$, then type "YES" and exit.
3. If $i > |FC(G')|$ and $j \leqslant |FC(G)|$, then type "NO" and exit.
4. If $j > |FC(G)|$, then type "YES" and exit.
5. If $FC(G')[i] \bmod FC(G)[j] = 0$, then $j = j + 1$.
6. $i = i + 1$ and go to step 2.

Obviously, the described algorithm allows us to find an answer, whether the colored graph G is included into the colored graph G'.

4. The search algorithm for all minimal edge k-extensions of an undirected colored graph

The previous sections 2 and 3 suggest two algorithms to cover canonical code generation and coloring preservation cases. Therefore we can formulate the algorithm of searching all minimal edge extensions based on the isomorphism rejection technique.

On the input, the algorithm takes $\mu_{\max}(G)$ of an undirected i-colored n-vertex graph G. The result is a set of maximal matrix codes for all non-isomorphic edge k-extensions of the defined graph. The algorithm offers the following steps to proceed.

1. Calculate G prime coloring code: $FC_{org} = FC(G)$.
2. Define a new graph as a copy of the original one: $G' = G$.
3. Append a new graph G' to a list of candidates CD: $CD \leftarrow G'$.
4. $ci = 0$.
5. If $|CD| = 0$, then exit.
6. $cd = CD_{ci}, ci = ci + 1$.
7. Calculate maximal matrix code for graph cd: $\mu_{\max}(cd)$.
8. Calculate new candidates CD_{new} from graph cd using the algorithm from section 2, which should take $\mu_{\max}(cd)$ and $f(cd)$ as input.
9. If $|CD_{new}| = 0$, then go to step 5.
10. Define set of extensions $EXT = \varnothing$.
11. $i = 0$.
12. Get i-th item from candidates list $G_i \in CD_{new}$ to verify whether the candidate is applicable to minimal edge extension requirements.
13. Iteratively remove each permutation of k edges from G_i.
14. For each G_i^* obtained by removing edge permutation calculate prime coloring code $FC(G_i^*)$ and run algorithm from section 3 to compare $FC(G_i^*)$ non-increasing ordered and FC_{org} non-increasing ordered.
15. If the algorithm's result is "YES", then G_i is a minimal edge k-extension, append it to EXT : $EXT \leftarrow G_i$.
16. $i = i + 1$.
17. If $i < |CD_{new}|$, then go to step 12.
18. If $|EXT| \neq 0$, then type EXT as a set of maximal matrix codes of all non-isomorphic minimal k-extensions of the defined undirected i-colored n-vertex graph and then exit.
19. Push items of CD_{new} to the end of CD : $CD \Leftarrow CD_{new}$ and go to step 5.

The algorithm accepts multithread implementation. It can be achieved by *producer-consumer* architecture, where CD contains required candidates, and different threads take it from CD and check whether the candidate is applicable for the resulting set of extensions.

5. Time estimation of the searching algorithm implementation

The resulting algorithm was implemented and tested on undirected graphs with the different numbers of vertices with 2 colors. The time estimation results are reflected in the Table. The graph's coloring is imagined as a vector of colors $(f(0), f(1), \ldots, f(n-1))$, where $f(v)$ stands for the color of vertex $v \in V$.

Table

The search algorithm's time estimation on undirected colored n-vertex graphs

n	Coloring vector	The amount of graphs	Total time	Average time, ms.
5	$(0,1,1,1,1)$	33	300 ms.	9
5	$(0,0,1,1,1)$	33	360 ms.	11
6	$(0,1,1,1,1,1)$	156	18 sec.	122
6	$(0,0,1,1,1,1)$	156	15 sec.	195
6	$(0,0,0,1,1,1)$	156	12 sec.	78
7	$(0,1,1,1,1,1,1)$	1044	7 m. 34 sec.	419
7	$(0,0,1,1,1,1,1)$	1044	7 m. 6 sec.	405
7	$(0,0,0,1,1,1,1)$	1044	6 m. 48 sec.	391
8	$(0,1,1,1,1,1,1,1)$	12345	2 h. 30 m. 2 sec.	743
8	$(0,0,1,1,1,1,1,1)$	12345	2 h. 12 m. 59 sec.	635
8	$(0,0,0,1,1,1,1,1)$	12345	2 h. 19 m. 33 sec.	657
8	$(0,0,0,0,1,1,1,1)$	12345	2 h. 5 m. 57 sec.	612

Conclusion

The investigation of the search problem for minimal edge extensions for undirected colored graphs has been done. The corresponding definitions have been considered in the context of the coloring function defined on undirected graphs. The article describes a modified canonical code method based on the isomorphism rejection technique. The pair of unique identificators have been found out for the case of non-isomorphic colored graphs. Moreover, there is an algorithm for a special unique way to generate new candidates from the defined canonical code. The study has proposed the search algorithm for all minimal edge extensions of the defined undirected colored graph. This algorithm has been implemented and tested for different graphs classes.

References

1. Avizienis A. Design of fault-tolerant computers. *AFIPS '67 (Fall): Proceedings of the November 14–16, 1967, fall joint computer conference*. New York, ACM, 1967, pp. 733–743. https://doi.org/10.1145/1465611.1465708
2. Hayes J. P. A graph model for fault-tolerant computing system. *IEEE Transactions on Computers*, 1976, vol. C-25, iss. 9, pp. 875–884. https://doi.org/10.1109/TC.1976.1674712
3. Harary F., Hayes J. P. Edge fault tolerance in graphs. *Networks*, 1993, vol. 23, pp. 135–142. https://doi.org/10.1002/net.3230230207
4. Harary F., Hayes J. P. Node fault tolerance in graphs. *Networks*, 1996, vol. 27, pp. 19–23. https://doi.org/10.1002/(SICI)1097-0037(199601)27:1%3C19::AID-NET2%3E3.0.CO;2-H

5. Abrosimov M. B. *Grafovye modeli otkazoustoichivosti* [Fault Tolerance Graph Models]. Saratov, Izd-vo Sarat. un-ta, 2012. 192 p. (in Russian).
6. McKay B. D. Isomorphism-free exhaustive generation. *Journal of Algorithms*, 1998, vol. 26, iss. 2, pp. 306–324. https://doi.org/10.1006/jagm.1997.0898
7. Brinkmann G. Isomorphism rejection in structure generation programs. *Discrete Mathematical Chemistry* (DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 51). 2000, pp. 25–38. https://doi.org/10.1090/dimacs/051/03
8. Abrosimov M. B., Sudani H. H. K., Lobov A. A. Construction of all minimal edge extensions of the graph with isomorphism rejection. *Izvestiya of Saratov University. Mathematics. Mechanics. Informatics*, 2020, vol. 20, iss. 1, pp. 105–115 (in Russian). https://doi.org/10.18500/1816-9791-2020-20-1-105-115
9. Abrosimov M. B., Kamil I. A. K., Lobov A. A. Construction of all nonisomorphic minimal vertex extensions of the graph by the method of canonical representatives. *Izvestiya of Saratov University. Mathematics. Mechanics. Informatics*, 2019, vol. 19, iss. 4, pp. 479–486 (in Russian). https://doi.org/10.18500/1816-9791-2019-19-4-479-486
10. Zykov A. A. *Osnovy teorii grafov* [The Basics of the Graph Theory]. Moscow, Vuzovskaya kniga, 2004. 664 p. (in Russian).

Поступила в редакцию / Received 25.01.2021
Принята к публикации / Accepted 29.04.2021
Опубликована / Published 31.08.2021