OVERVIEW OF GRAPH COLORING METHODS AND ALGORITHMS

Oleksii Denysenko
Software developer
SP Denysenko
UKRAINE

OVERVIEW OF GRAPH COLORING METHODS AND ALGORITHMS

ΟΓΛΥΑΔ ΜΕΤΟΔΙΩΝ ΤΑ ΑΛΓΟΡΙΤΜΙΩΝ ΡΟΖΦΑΡΒΟΒΩΝΑΝΑ ΧΡΑΓΦΥ

АНАТОЦЯ | ABSTRACT:
The problem of graph coloring using heuristic methods was considered. The purpose of the work is to analyze heuristic methods and describe computational experiments using a program for heuristic evaluation of a chromatic number. Such methods as the greedy algorithm, the full-search method, the random-search method, and the depth-limited search method were considered. Experiments aimed at evaluating the quality of the solutions formed by different methods were conducted. Comparison of the results of the use of different methods for graphs which differ in such properties as, for example, the number of vertices and connection density, depending on search parameters of specific methods, gives us data for the analysis of the relevance of the use of these methods for the solution of specific practical problems, such as scheduling, cluster analysis, calculation of derivatives, parallelization of numerical methods, frequency distribution, digital signs, allocation of processor registers.

FORMULATION OF THE PROBLEM.
The task of finding the optimal graph coloring is one of the main problems of graph theory.

There are currently various methods and algorithms for solving this problem, which differ in their computational complexity, which limits their use for graphs that meet certain criteria, as well as in the probability of obtaining the optimal result [1-5].

Requirements for methods of finding the optimal coloring may vary for different tasks, depending on which parameters are more important – the time of the method’s operation or the probability of obtaining the optimal result.

This explains variety of existing search methods, each of which has its own advantages that make it use relevant to the particular situation.

THE PURPOSE OF THE ARTICLE.
Comparison of the results of the use of different methods for graphs which differ in such properties as, for example, the number of vertices and connection density, depending on search parameters of specific methods, gives us data for the analysis of the relevance of the use of these methods for the solution of specific practical problems [6-10].

THE MAIN MATERIAL.
Theoretical part. One of the problems of the graph theory is the problem of finding minimum coloring of an undirected graph

\[ G = (A, V) \]

where:

\[ A = \{a_1, a_2, \ldots, a_N\}, \quad |A| = N \text{ is a vertex-set}, \]

\[ V = \{v_1, v_2, \ldots, v_M\}, \quad |V| = M \text{ is an edge set}. \]

The minimum number of colors required to color a given graph is called achromatic number. The set task belongs to the NP class and does not allow one to find the optimal solution (or, depending on the formulation, the exact value of the chromatic number) in polynomial time (under the condition P≠NP), so in practice various heuristic algorithms are used in order to solve it [12].

Chromatic number of a graph is a minimum k number where vertex-set V can be divided into k non-overlapping classes, while ensuring that vertices in each class are independent, that is, any graph edge does
not connect vertices of the same class.

A K-colored graph is a graph whose chromatic number does not exceed K. That is, its vertices can be colored with K different colors in such a way that the ends of any edge will have different colors.

**Practical application.** Many in-depth problems of the graph theory are easily stated in coloring terms. The most famous of these problems, the four-color problem, is now solved, but new ones appear, for example, generalization of the four-color problem, the Hadwiger’s conjecture [13].

In problems related to scheduling theory, inspections are time intervals. Every inspection can be considered by matching it with a graph vertex; it is worth noting that two arbitrary vertices of the graph are connected by an edge, if their corresponding inspections cannot be implemented at one moment in time. The task of scheduling inspections, while ensuring that the least time

Coloring means comparison of numerical estimates (“colors”) with c(a) vertices of the graph, which ensures that no pair of adjacent vertices has matching colors [11].

Correct k-coloring of a G graph with k colors is a partition of the set V of its vertices into k disjoint subsets is spent, is equivalent to the task of finding minimum coloring of a graph. Chromatic number of a graph will actually be equivalent to the least time-consuming inspection [14].

Suppose in order to perform some n jobs, you need to allocate m available resources. We assume that each of the jobs is performed during some uniform period of time, and that a subset of S resources is required to complete the i-th job.

Let us build a graph G: every job corresponds to a vertex of the graph, and an edge (x, x) exists in the graph when at least one common resource is required to execute the i-th and j-th jobs. Graph coloring determines allocation of resources where jobs that correspond to vertices colored with a color of the same number are performed at the same time. The most rational use of resources (execution of jobs in the least time) corresponds to the optimal coloring of graph G.

Graph coloring is used in practice for:
- scheduling;
- cluster analysis;
- calculation of derivatives;
- parallelization of numerical methods;
- frequency distribution;
- digital watermarks;
- allocation of processor registers;
- designing devices in which wires, connected in one node, should have different colors for convenience of distinguishing [15].

**Comparison of algorithms for finding chromatic number of a graph.** The graph coloring problem is an NP-complete problem. In the basic case, chromatic number of a graph cannot be calculated using only its standard numerical characteristics, such as the number of vertices, edges, connected components, and distribution of vertex degrees.

As noted above, the set task belongs to the NP class and does not allow one to find the optimal solution (or, depending on the formulation, the exact value of the chromatic number) in polynomial time (under the condition P≠NP), so in practice various heuristic algorithms are used in order to solve it.

There are various algorithms for finding chromatic number of a graph, which differ in the accuracy of the chromatic number of a graph and the time in which this number can be found [16].

The most accurate (closest to optimal) result is given by the exhaustive search algorithm, which belongs to the class of methods for finding a solution by exhausting all possible options. Complexity of the exhaustive search method depends on the number of all possible solutions to the problem.

It is possible to slightly reduce the time of exhaustive search by eliminating all possible search options which obviously are not a solution from the set. This is achieved with the help of lower and upper bounds of chromatic number of a graph or by preliminary calculation of coloring close to the optimal one by using methods that give rougher and more inaccurate results but do not take a lot of time.

Other algorithms that are not associated with the search though a large number of colorings, as a rule, give a less accurate (less close to optimal) result, but in much less time.

Examples of other algorithms:
1. Greedy algorithm – first the first vertex is colored with the minimum possible color, then the second one, the third one, and so on are colored until all the vertices are colored.
2. The vertex with the greatest number of arcs is found and colored with color 1. One of the neighboring vertices is taken and colored with color 2, another neighboring one is colored with the minimum color of its neighboring one plus 1.
3. All vertices are sorted in descending order by the number of arcs, and coloring starts from the vertex with the maximum number of arcs.
4. With the help of the greedy algorithm chromatic number of a graph is found. All the options that are «the worst» in comparison with the greedy algorithm are removed from the many possible graph coloring options.
The optimal solution is found from the remaining set using the search method. The number of discarded options will be equal to: \( N^N - g^N \), where \( g \) is the chromatic number of a given graph with \( N \) vertices, obtained using the greedy algorithm.

The greedy algorithm. The greedy algorithm is the simplest method for finding coloring of an undirected graph, which would be close to a chromatic number. The greedy algorithm consists in sequential coloring of all \( X \) vertices of the graph \( G \) with the minimum possible color, provided that no vertex adjacent to the given one has been colored with this color. As for the minimum graph coloring problem, this method does not guarantee that the optimal solution will be found. The time complexity of the method is linear in the number of vertices (\( n \)) in a graph – \( O(n) \).

Being the fastest, this method can give a result that is far from optimal (where coloring of the graph would be minimum).

There can be two software implementations of the greedy approach strategy, which are not based on vertex degree consideration:
- coloring of vertices in random order;
- recursive coloring of vertices: an arbitrary vertex of a graph is colored with a minimum possible color, and then the method is recursively applied to every vertex adjacent to the current one.

Since the greedy algorithm does not provide optimal coloring, various strategies based on vertex degree consideration are used when organizing sequential coloring of graphs for the purpose of approximating the results given by the greedy algorithm to the optimal one. For example, the strategy which presupposes search for vertices with the largest number of arcs that (and adjacent thereto) will be colored first [17].

Another strategy presupposes preliminary sorting of all vertices of a graph by non-increase of their degrees and their subsequent coloring.

Computational complexity of such an algorithm is greater than the complexity of other greedy search methods, since preliminary sorting of graph vertices is required. In Figure 1 you can see the sequence of vertices coloring in case of the use of a greedy algorithm. In Figure 2 you can see the sequence of vertices coloring when finding chromatic number with the use of a recursive variation of the greedy algorithm.

Exhaustive search. The most accurate result is provided by the exhaustive search algorithm, which belongs to the class of methods for finding a solution by exhausting all possible options. Complexity of exhaustive search depends on the number of all possible solutions to the problem. If the decision space is very large, exhaustive search may not produce results in a long period of time. For example, for a graph with \( N \) vertices, the number of options considered will be \( N^N \).

There can be two software implementations of exhaustive search. The first one presupposes recursive traverse of all graph vertices and search through all possible coloring options for every vertex and those adjacent thereto.

The other one presupposes sequential generation of coloring for all graph vertices and checking whether they are optimal.

Both approaches give an optimal result, but recursive search allows one to do this in a relatively longer time.

In both approaches, it is possible to use a reference solution obtained with the help of the fast-greedy algorithm, which is used to reduce the number of considered options, which, as a result, leads to a decrease in the operating time of this method [18].

Thus, complexity of the algorithm which presupposes the use of a reference solution is:

\[ O(gN) \]
where \( N \) is the number of graph vertices, and \( g \) is the result obtained with the help of the greedy method.

But the advantage in time, which the algorithms that use reference solution have, are only significant for relatively small \( N \) (about 10-20 vertices).

**Random-search method.** The random-search method consists in random search through a given number of options (\( Z \)) for graph coloring and selection of the most optimal one. In Figure 3 you can see an example of the use of random-search method [18].

Algorithm complexity: \( O(Z) \).

![Fig. 3. Example of the use of random-search method](image)

**Depth-limited search method.** The method consists in recursive traverse of all vertices of a graph with a predetermined depth traversal limit \( l \). The algorithm of this method is presented below.

Initially, all vertices are considered open (placed in the Open array).

Let us calculate the reference solution using the fast method and store it in best Solution, and the number of colors will be store in lucky Nums.

1) Then we choose the next vertex. And call it \( i \).
2) Let us assume that next Color := 0.
3) Let us choose the next color next Color.
If it is not less than lucky Nums, go to step 1.
4) Then we color vertex \( i \) with next Color color.
5) Then we increment the current level of nesting deep Level. If deep Level is greater than the predetermined deep Limit, then go to step 3, having decremented deep Level first.
6) Then we put \( i \) into Open.
7) Let us select the next vertex \( j \), adjacent to \( i \), which, at the same time, belongs to the Open array [19].

In Figure 4 1-3 you can see an example of graph coloring with the help of depth-limited search for values that correspond to the depth limit. \( 1=0, 1=1, 1=2 \).

Time complexity \( O(b^{d+1}) \) where \( b \) is branching factor, and \( d \) is search depth.

![Fig. 4. Depth-limited search method](image)

**Experimental studies.** To carry out computational experiments, we used a program developed for heuristic evaluation of chromatic number. The results of experiments are presented in tables 1-4.

**Table 1**

<table>
<thead>
<tr>
<th>Density</th>
<th>0,1</th>
<th>0,5</th>
<th>0,1</th>
<th>0,5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of vertices</td>
<td>100</td>
<td>100</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>Перебор случайный</td>
<td>1,088</td>
<td>1,084</td>
<td>1,123</td>
<td>1,049</td>
</tr>
<tr>
<td>Greedy recursive</td>
<td>1,082</td>
<td>1,081</td>
<td>1,099</td>
<td>1,051</td>
</tr>
<tr>
<td>Greedy by clicks</td>
<td>1,086</td>
<td>1,059</td>
<td>1,11</td>
<td>1,056</td>
</tr>
<tr>
<td>Greedy</td>
<td>1,088</td>
<td>1,084</td>
<td>1,123</td>
<td>1,049</td>
</tr>
<tr>
<td>Density</td>
<td>0,5</td>
<td>0,9</td>
<td>0,5</td>
<td>0,9</td>
</tr>
<tr>
<td>Number of vertices</td>
<td>300</td>
<td>300</td>
<td>400</td>
<td>400</td>
</tr>
<tr>
<td>Random search</td>
<td>1,045</td>
<td>1,047</td>
<td>1,0475</td>
<td>1,036</td>
</tr>
<tr>
<td>Greedy recursive</td>
<td>1,058</td>
<td>1,046</td>
<td>1,045</td>
<td>1,039</td>
</tr>
<tr>
<td>Greedy by clicks</td>
<td>1,060</td>
<td>1,038</td>
<td>1,045</td>
<td>1,039</td>
</tr>
<tr>
<td>Greedy</td>
<td>1,045</td>
<td>1,046</td>
<td>1,047</td>
<td>1,036</td>
</tr>
</tbody>
</table>
Table 2
Confidence intervals for average values of chromatic number of the graph for a sample of 100 graphs

<table>
<thead>
<tr>
<th>Number of vertices</th>
<th>Arc probability</th>
<th>Random search</th>
<th>Greedy recursive</th>
<th>Sorting of vertices</th>
<th>Coloring of vertices with the largest degrees</th>
<th>Greedy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Dispersion: 1,28</td>
<td>Interval: [14,91; 15,36] ε: 0,225</td>
<td>Dispersion: 0,28</td>
<td>Interval: [14,7 ;14, 9] ε: 0,105</td>
<td>Dispersion: 0,26</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Interval: [13,45; 13,66] ε: 0,102</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3
Confidence intervals for average values of chromatic number of the graph for a sample of 100 graphs

<table>
<thead>
<tr>
<th>Number of vertices</th>
<th>Arc probability</th>
<th>Random search</th>
<th>Greedy recursive</th>
<th>Sorting of vertices</th>
<th>Coloring of vertices with the largest degrees</th>
<th>Greedy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Dispersion: 1,399</td>
<td>Interval: [20,75; 21,22] ε: 0,23</td>
<td>Dispersion: 0,3676</td>
<td>Interval: [20,69 ;20,94] ε: 0,12</td>
<td>Dispersion: 0,5636</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Interval: [19,27; 16,56] ε: 0,14</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4
Comparison of time required for chromatic number search methods versus the recursive greedy method

<table>
<thead>
<tr>
<th>Arc probability</th>
<th>Number of vertices</th>
<th>Random search</th>
<th>Greedy by clicks</th>
<th>Greedy sorting</th>
<th>Greedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>100</td>
<td>14609</td>
<td>11</td>
<td>14</td>
<td>2</td>
</tr>
<tr>
<td>0.5</td>
<td>100</td>
<td>3181</td>
<td>4</td>
<td>5</td>
<td>1,1</td>
</tr>
<tr>
<td>0.9</td>
<td>100</td>
<td>1304</td>
<td>4</td>
<td>7</td>
<td>0,8</td>
</tr>
<tr>
<td>0.1</td>
<td>200</td>
<td>7806</td>
<td>11</td>
<td>4</td>
<td>1,8</td>
</tr>
<tr>
<td>0.5</td>
<td>200</td>
<td>1299</td>
<td>5</td>
<td>5</td>
<td>1,2</td>
</tr>
<tr>
<td>0.9</td>
<td>200</td>
<td>493</td>
<td>3</td>
<td>3</td>
<td>0,9</td>
</tr>
</tbody>
</table>

In practice it is impossible to obtain the optimal result for graphs with more than 10-13 vertices with the help of exhaustive search, even taking into account the use of upper and lower bounds. Out of all these algorithms, the result which is the closest to the optimal one is given by the method which presupposes sorting of vertices by non-increase of their degrees, which, despite not being the fastest one, still solves the set task in an acceptable period of time, even in case of a large number of vertices.

CONCLUSIONS AND SUGGESTIONS.

In the article we considered algorithms for finding optimal graph coloring and carried out experiments with various input parameters. Based on the results obtained, a comparative analysis of the algorithms was carried out, on the basis of which a conclusion on the feasibility of their use in practice was drawn.

References:
Автор(и):

Денисенко О.О., разробник програмного обеспечения
ФОП Денисенко
УКРАЇНА

© Denysenko O., 2019