**International Journal of Electrical and Computer Engineering (IJECE)**
Vol. 15, No. 5, October 2025, pp. 4714~4722
ISSN: 2088-8708, DOI: 10.11591/ijece.v15i5.pp4714-4722

# Influence of the graph density on approximate algorithms for the graph vertex coloring problem

**Velin Kralev, Radoslava Kraleva**
Department of Informatics, South-West University “Neofit Rilski”, Blagoevgrad, Bulgaria

**Article Info**

***Article history:***

Received Aug 14, 2024
Revised Apr 14, 2025
Accepted Jul 3, 2025

***Keywords:***

Chromatic number
Graph coloring
Graph density
Graph theory
Greedy algorithm

**ABSTRACT**

This research explores two heuristic algorithms designed to efficiently solve the graph coloring problem. The implementation codes for both algorithms are provided for better understanding and practical application. The experimental methodology is thoroughly discussed to ensure clarity and reproducibility. The execution times of the algorithms were measured by running the test applications six times for each analyzed graph. The results indicate that the first algorithm generally produced better solutions than the second. In only two instances did the first algorithm produce solutions comparable to those of the second. The results reveal another trend: as the graph density exceeds 85%, the number of required colors increases significantly for both algorithms. However, even at a density of 95%, the number of colors required to color the graph's vertices does not exceed half the total number of vertices. As the graph density increases from 95% to 100%, the number of colors required to color the graph rises significantly. However, when the graph density exceeds 97%, both algorithms produce identical solutions.

***Corresponding Author:***

Velin Kralev
Department of Informatics, Faculty of Mathematics and Natural Science, South-West University
66 Ivan Michailov str., 2700 Blagoevgrad, Bulgaria
Email: velin_kralev@swu.bg

## 1. INTRODUCTION

Informally, the labeling (or coloring) of a graph's vertices can be described as assigning each vertex a specific label (or color). The goal of this process is to assign labels (colors) such that no two adjacent vertices share the same label (color). The labels (or colors) assigned to the vertices are elements of a finite set, with each vertex receiving exactly one of these elements. A collection of vertices assigned the same label (color) is known as a chromatic (or color) class. If *c*-number sets are formed, *i.e.*, classes can define a graph as *c*-colorable. Natural numbers from 1 through *c* are usually used to denote these chromatic classes. For a graph coloring to be valid, every pair of adjacent vertices (*i.e.*, vertices connected by an edge) must be assigned different labels (colors). Formally and in general, it can be defined that when a graph is colored acceptably (correctly) for example with *c* colors, then it is *c*-colorable. An acceptable coloring of a graph always exists. If a graph has *n* vertices, each vertex can be assigned a unique color, resulting in exactly *n* chromatic classes. This will certainly be an acceptable labeling (or coloring) of the given graph. With this coloring scheme, no two vertices will share the same label (or color), regardless of whether they are connected by an edge [1], [2].

The smallest possible number of chromatic classes in which the vertices of a given graph can be distributed is called *the optimal coloring of the graph* and is denoted by $\chi(G)$. If a graph is not complete then $\chi(G)$ is certainly less than the number of vertices of the given graph, *i.e.*, $\chi(G) < n$. If it is found that for a graph G, χ(G)=*c*, then this graph is *c*-chromatic. The correct (acceptable) coloring of a *c*-color graph is the grouping of the vertices of that graph into *c* sets, the vertices in each of these sets being unrelated to each other but may be (and usually are) connected vertices of different sets. If graph G' is a subgraph of a graph G, then any acceptable coloring of the graph G will also be an acceptable coloring of the graph G'. Moreover, the chromatic index of a graph G' is less than or at most equal to the chromatic index of a graph G [3], [4].

The problem of labeling (coloring) vertices (in the field of graph theory) is an NP-hard problem [5] and is still actively studied [6]–[9]. Various approaches, methods, and algorithms for solving this problem continue to be actively researched in scientific literature. For instance, the maximal independent set for the vertex-coloring problem on planar graphs [10], the adjacent vertex-distinguishing edge coloring [11], the rainbow vertex coloring problem [12], and many others. Different methods of the problem use various algorithms [13], [14], approaches [15]–[17] and techniques [18], [19]. Other methods are used to find a solution to such problems in the field of graph theory [20]. Many other more detailed and in-depth analyzes of this problem are discussed in [21]–[23].

Any complete graph that does not contain parallel edges and loops can be colored with exactly |V| of number of colors, where |V| is the number of vertices in this graph. This is because in every complete graph for every pair of vertices there is an edge that connects these vertices. Also, in every complete graph, every vertex is connected (by an edge) to every other vertex. Therefore, the chromatic index of any complete graph is equal to the number of vertices in that graph. Because of this statement, it can be concluded that if a graph has its complete subgraph, then the chromatic index of this graph will be at least equal (or greater) to the number of vertices that form the complete subgraph in the given graph. It is also known that in each graph that has its complete subgraph, it is possible for the chromatic index of this graph to be greater (but not less) than the number of vertices forming the complete subgraph [24].

In this paper, some results obtained after the execution of two modified versions of the greedy algorithm used to solve the problem of labeling (coloring) the vertices of a graph – Greedy coloring algorithm (GCA) [25] are presented and analyzed. The first version of the greedy algorithm is classical and the initial arrangement of the vertices does not change when it is executed. With this modification, the vertices are labeled (colored) in the order in which they were added to the graph - *i.e.*, by index (greedy coloring algorithm based on the index ordering – GC-IDX). The second variant of the greedy algorithm is implemented as the initial order of the vertices is changed, but in a random arrangement way, *i.e.* a random arrangement of vertices is generated, one from all possible ones (which are exactly n!). After this step, the vertices of the graph are labeled (colored) according to the generated order of vertices – *i.e.*, randomly (greedy coloring algorithm based on the random ordering - GC-RND). Both variants of the greedy algorithm are approximate, and thus, it is not guaranteed that either modification will find the optimal solution for labeling (coloring) the graph's vertices using the fewest labels (colors) possible. When solving NP-hard problems with large input data, only approximate algorithms can be used, since they can generate at least a close to the optimal solution and at an acceptable time, for example in the order of seconds to few minutes. Other algorithms for solving specific variants of the graph vertex labeling (coloring) problem are also known and discussed in other publications [26], [27].

## 2. RESEARCH METHOD

This section shows and discusses the source codes of both algorithms GC-IDX and GC-RND algorithms. Both algorithms are approximate and solve the graph vertex coloring problem approximately. For both algorithms – GC-IDX and GC-RND, some parameters, dynamic arrays and matrices need to be declared in advance. They are presented in Figure 1.

```
var
  NumberOfVertices: Integer;
  ArrayOfColors: array of TColor;
  MinimalColors, RandomCount: Integer;
  BestMinimalColors, BestIteration: Integer;
  AdjacencyMatrix: array of array of Integer;
```

Figure 1. Source code of preliminary declarations

The *NumberOfVertices* parameter stores the number of vertices in the graph. The *ArrayOfColors* array is used by both algorithms. Each item of this dynamic structure contains an item of type *TColor* with which the given vertex of the graph is colored. Both algorithms (CG-IDX and CG-RND) change the values of these items. The *MinimalColors* parameter is aggregated and is used by the algorithms in the solution search loop. Each graph is represented by an adjacency matrix (line 6), which is used by both algorithms. Each item [*u*, *v*] of the matrix shows whether vertices with indices u and v are adjacent or not.

The *GreedyColoringIDX* procedure implements the first approximate method for labeling (coloring) graph vertices. The source code of this procedure is shown in Figure 2. This procedure also uses some additional parameters – *CurrentColor*, *V*, *Vertex* and *AcceptableSolution*. The *CurrentColor* parameter holds the number of one of the colors used. Parameter *V* is used when iterating the adjacency matrix when searching for the neighbors of the given vertex. The *AcceptableSolution* parameter indicates whether the given vertex has been successfully colored with any of the available colors.

```
01 procedure GreedyColoringIDX;
02 begin
03   var AcceptableSolution: Boolean := False;
04   var V := 0; var Vertex := 0; var CurrentColor := 0;
05   MinimalColors := 0;
06   for Vertex := 1 to NumberOfVertices do
07   begin
08     CurrentColor := 0;
09     while not AcceptableSolution do
10     begin
11       CurrentColor := CurrentColor + 1;
12       AcceptableSolution := True;
13       for V := 1 to NumberOfVertices do
14       begin
15         if ((AdjacencyMatrix[Vertex][V] > 0) and
16             (ArrayOfColors[V]=CurrentColor)) then
17         begin
18           AcceptableSolution := False;
19           Break;
20         end;
21       end;
22     end;
23     ArrayOfColors[Vertex] := CurrentColor;
24     if (MinimalColors < CurrentColor) then
25       MinimalColors := CurrentColor;
26   end;
27 end;
```

Figure 2. Source code of the *GreedyColoringIDX* procedure

The parameters *MinimalColors*, *V*, *Vertex* and *CurrentColor* are set to 0 in the beginning of the *GreedyColoringIDX* procedure (rows 4 and 5). The procedure checks which of the current colors can be used to color the given vertex (through a for-loop started on row 6). Finding the color with the smallest possible index to be used for coloring is realized by a while-loop (rows 09 - 22). Just before the execution of the loop, the *CurrentColor* parameter is set to 0 (row 08). In line 11, the *CurrentColor* parameter is incremented by 1, which means that on the first iteration of the loop, this parameter will be set to 1. Through the *for* loop (executed between rows 11-21), it is checked whether any of the neighboring (adjacent) vertices of the current vertex (parameter *Vertex*) is not already colored with the color *CurrentColor*. If this is not the case, then the current vertex is colored with *CurrentColor*. The execution of the *for* loop can be prematurely terminated if a vertex which is adjacent to the current one and is already colored with the *CurrentColor* is found. If this happens, the *AcceptableSolution* parameter is set to *False* before the loop is terminated. In this situation, the coloring of the current vertex with the *CurrentColor* is impossible, and the procedure executes a new iteration of the *while* loop and increasing the index of the current color by 1 (row 11). Execution of the *while* loop (rows 09 - 22) continues until an acceptable (possible) color is found for the current vertex. In this case, the parameter *AcceptableSolution* will be set to *True* (on row 12 before starting the *for* loop). The value of the *CurrentColor* parameter is stored in the array *ArrayOfColors* in the item associated with the parameter *Vertex* (row 23). The value of the *CurrentColor* parameter is copied to the *MinimalColors* parameter only if the value of the *CurrentColor* parameter is greater than the value of the *MinimalColors* parameter.

When the value of the *MinimalColors* parameter is increased by one, it means that the available colors are not enough to color the current vertex and a new color needs to be added, which is done by increasing by one the value of the *MinimalColors* parameter. The execution of the *for* loop (lines 06 – 26) ends only when all the vertices in the graph are colored, and in such a way that no pair of adjacent vertices are colored with the same color. After the execution of the *GreedyColoringIDX* procedure, the *MinimalColors* parameter stores the minimum number of colors needed to color the given graph, according to the implementation of the algorithm constructed in this way.

The *GreedyColoringRND* procedure implements the second approximate method for coloring graph vertices. The source code of this procedure is shown in Figure 3. This procedure also uses some additional parameters – *BestMinimalColors*, *BestIteration* and *RandomCount*. The *BestMinimalColors* parameter holds the minimal number of colors needed to color the graph vertices. This parameter has a different purpose than the *MinimalColors* parameter, which contains the minimum number of colors required when running the current *GreedyColoringIDX* procedure. In contrast to this parameter, the *BestMinimalColors* parameter contains the minimum number of colors to color the graph after executing several *GreedyColoringIDX* procedures. In contrast to this parameter, the *BestMinimalColors* parameter contains the minimum number of colors to color the graph after running the *GreedyColoringIDX* procedure several times. The number of these calls is determined by the *RandomCount* parameter, which is initialized when the *GreedyColoringRND* procedure is started. Accordingly, the *BestIteration* parameter stores which call the *GreedyColoringIDX* procedure found the best solution, information about which is stored in the *BestMinimalColors* parameter.

The essence of the *GreedyColoringRND* procedure consists in the successive generation of different (random) permutations of the vertices of the graph, based on which, in the next step, these vertices will be colored according to this order by the *GreedyColoringIDX* procedure. The random order of the vertices is generated by the *for* loop (lines 07 – 12), starting from the last vertex for which a new index of another vertex is chosen (randomly) with which the current one is exchanged (line 11). In the next step, a new (random) index is chosen for the penultimate vertex, which is exchanged with some vertex from those before it. This process is repeated until all vertices up to the first have been randomly swapped. After the new order is generated, the procedure *GreedyColoringIDX* is called, which colors the vertices of the graph in the thus generated suborder. When the *GreedyColoringIDX* procedure is executed, the number of colors needed to color all vertices of the graph is calculated. This count is stored in the *MinimalColors* parameter, which is compared to the best solution found from a previous iteration of the algorithm. If the current solution is better (*i.e.* the *MinimalColors* parameter has a smaller value than the *BestMinimalColors* parameter) then the current solution is stored as the best found so far. Accordingly, the *BestIteration* parameter stores the iteration at which this (best) solution was found.

```
01 procedure GreedyColoringRND;
02 begin
03   BestMinimalColors := MaxInt; RandomCount := 100;
04   BestIteration := 0;
05   for var Iteration := 1 to RandomCount do
06   begin
07     for var V := NumberOfVertices downto 1 do
08     begin
09       var I := V;
10       var J := Random(V) + 1;
11       var T := I; I := J; J := T; // Swap vertices I and J
12     end;
13     GreedyColoringIDX();
14     if MinimalColors < BestMinimalColors then
15     begin
16       BestMinimalColors := MinimalColors;
17       BestIteration := Iteration;
18     end;
19   end;
20 end;
```

Figure 3. Source code of the *GreedyColoringRND* procedure

## 3. RESULTS AND DISCUSSION

The results of the experiment will be presented. An analysis between the two approximate methods will also be presented in terms of the number of minimum colors required to color the test graphs, as well as the running time of the algorithms. For this study, 24 graphs with 1000 vertices and different numbers of edges were created. These graphs are undirected and contain no multi-edges (parallel edges or multiple edges) or loops. The number of edges is determined by the graph density (in percentage) that is set for each graph. This parameter ranges from 5 to 95 for graphs G1 – G19 and changes in 5 percent increments. For graphs G20 – G24 this parameter ranges from 96 to 100 and changes in 1 percent increments.

Each graph contains the same number of vertices (1000). The maximum possible number of edges that a graph with 1000 vertices can contain (without parallel edges and loops) is 1000*(1000-1)/2=499500. From these edges, a certain percentage of them is randomly selected based on the graph density parameter. Additional information about the test graphs and the results of the execution of the two algorithms are shown in Tables 1 and 2. The experimental conditions are: 64-bit Win 11 OS and hardware configuration: Processor: Intel (R) Core (TM) i5-12450H at 4.40 GHz; RAM: 16 GB, SSD 1000 GB.

Table 1. Results of the approximate algorithms for graphs G1 – G19

| Graph number | Graph file name | Density (%) | Edges (count) | Greedy Coloring (IDX) Colors | Greedy Coloring (IDX) Time (ms) | Greedy Coloring (RND) Colors | Greedy Coloring (RND) Time (ms) | Greedy Coloring (RND) Iteration | Diff in colors |
|---|---|---|---|---|---|---|---|---|---|
| 1 | G_1000_24976 | 5 | 24 976 | 15 | 31 | 19 | 2609 | 17 | 4 |
| 2 | G_1000_49950 | 10 | 49 950 | 25 | 47 | 31 | 2625 | 126 | 6 |
| 3 | G_1000_74926 | 15 | 74 926 | 35 | 63 | 41 | 2656 | 42 | 6 |
| 4 | G_1000_99900 | 20 | 99 900 | 45 | 76 | 53 | 2672 | 131 | 8 |
| 5 | G_1000_124876 | 25 | 124 876 | 55 | 78 | 66 | 2688 | 209 | 11 |
| 6 | G_1000_149850 | 30 | 149 850 | 66 | 79 | 75 | 2703 | 153 | 9 |
| 7 | G_1000_174826 | 35 | 174 826 | 77 | 81 | 85 | 2718 | 127 | 8 |
| 8 | G_1000_199800 | 40 | 199 800 | 86 | 93 | 94 | 2734 | 72 | 8 |
| 9 | G_1000_224776 | 45 | 224 776 | 99 | 109 | 108 | 2735 | 91 | 9 |
| 10 | G_1000_249750 | 50 | 249 750 | 113 | 110 | 124 | 2750 | 103 | 11 |
| 11 | G_1000_274726 | 55 | 274 726 | 128 | 112 | 139 | 2751 | 24 | 11 |
| 12 | G_1000_299700 | 60 | 299 700 | 141 | 125 | 152 | 2797 | 197 | 11 |
| 13 | G_1000_324676 | 65 | 324 676 | 157 | 127 | 169 | 2798 | 46 | 12 |
| 14 | G_1000_349650 | 70 | 349 650 | 174 | 156 | 175 | 2813 | 81 | 1 |
| 15 | G_1000_374626 | 75 | 374 626 | 195 | 157 | 204 | 2828 | 28 | 9 |
| 16 | G_1000_399600 | 80 | 399 600 | 222 | 172 | 226 | 2875 | 73 | 4 |
| 17 | G_1000_424576 | 85 | 424 576 | 255 | 188 | 256 | 2905 | 62 | 1 |
| 18 | G_1000_449550 | 90 | 449 550 | 293 | 203 | 301 | 2906 | 158 | 8 |
| 19 | G_1000_474526 | 95 | 474 526 | 384 | 248 | 394 | 2922 | 94 | 10 |

Table 2. Results of the approximate algorithms for graphs G20 – G24

| Graph number | Graph file name | Density (%) | Edges (count) | Greedy Coloring (IDX) Colors | Greedy Coloring (IDX) Time (ms) | Greedy Coloring (RND) Colors | Greedy Coloring (RND) Time (ms) | Greedy Coloring (RND) Iteration | Diff in colors |
|---|---|---|---|---|---|---|---|---|---|
| 20 | G_1000_479520 | 96 | 479 520 | 419 | 265 | 422 | 2975 | 65 | 3 |
| 21 | G_1000_484516 | 97 | 484 516 | 461 | 287 | 460 | 3052 | 27 | 1 |
| 22 | G_1000_489510 | 98 | 489 510 | 856 | 304 | 856 | 3196 | 41 | 0 |
| 23 | G_1000_494506 | 99 | 494 506 | 902 | 352 | 902 | 3351 | 34 | 0 |
| 24 | G_1000_499500 | 100 | 499 500 | 1000 | 407 | 1000 | 3828 | 79 | 0 |

In Tables 1 and 2, the "Graph number" column shows the number of the test graph. The "Graph file name" column shows the name of the file where the information for the corresponding test graph is stored. The "Density (%)" column shows the density of the test graph in terms of the number of edges that this graph contains out of all the possible edges that this graph would have if it were complete. Accordingly, the column "Edge count" shows the number of these edges. The "Color" columns show the number of required colors that each of the algorithms used to color the corresponding graph. Accordingly, the "Time (ms)" columns show the time for each of the two algorithms to generate a solution. The "Iteration" column shows the best iteration of the GC-RND algorithm where the algorithm generated the best solution out of a total of 250 iterations performed. The "Diff in colors" column shows the difference in chromatic classes between the GC-RND algorithm and the CG-IDX algorithm.

Table 1 and the charts in Figures 4 and 5 show the results of the two algorithms in terms of the number of required colors that the two algorithms used to color the graphs from G1 through G19. The results also show that for all nineteen graphs from G1 through G19, the GC-IDX algorithm found better solutions than the GC-RND algorithm. In two cases, only the number of chromatic classes (colors) generated by the GC-RND algorithm are close to those generated by the GC-IDX algorithm – these are the cases for graphs G14 and G17. For graph G14, the GC-IDX algorithm generated a solution where 174 colors were needed to color the graph. Accordingly, the GC-RND algorithm found a very close solution, where 175 colors were needed to color the graph (*i.e.*, only one color more than the colors the GC-IDX algorithm used). In all other cases, the GC-IDX algorithm found solutions that were better than those found by the GC-RND algorithm. The difference in the number of colors varies between 4 and 12, with graphs G1 and G16 having a difference of 4 colors and graph G13 having a difference of 12 colors as shown in Figure 5. However, Figures 4 and 5 show that as the number of edges in the graph increases, which is a consequence of the graph density increasing, the GC-RND algorithm begins to generate solutions that are closer to those generated by GC-IDX algorithm. From the trend of the obtained results, it can be concluded that for graphs with a higher density, for example greater than 70%, the GC-RND algorithm can be successfully used to generate solutions.

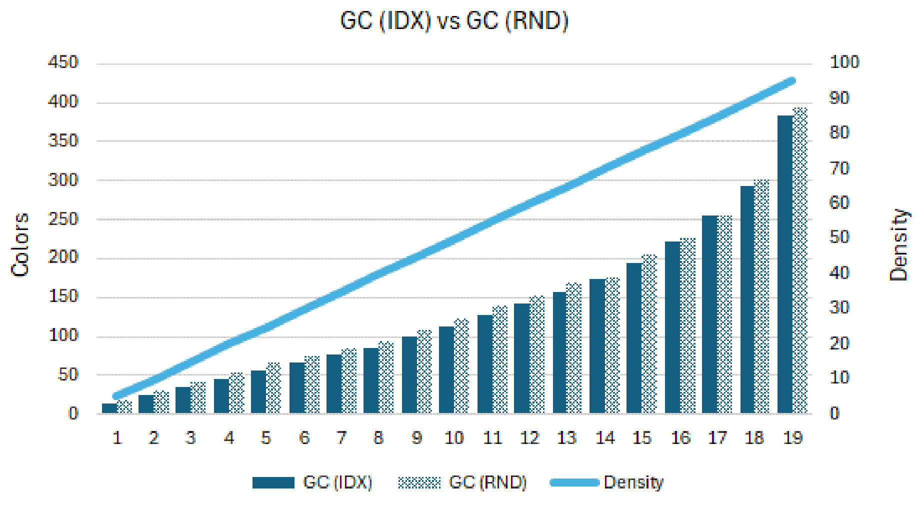

Figure 4. The number of colors (left y-axis) generated by the two algorithms at different graph densities (right y-axis)

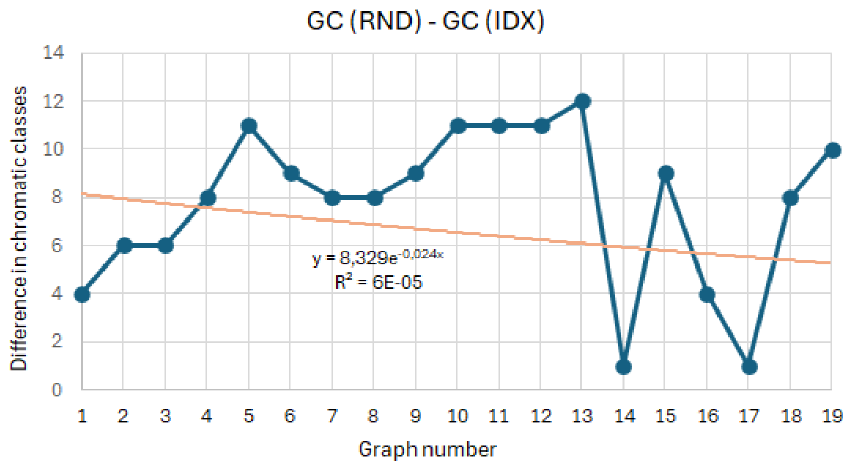

Figure 5. Difference in chromatic classes between the GC (RND) algorithm and the CG (IDX) algorithm

The chart in Figure 4 shows another trend, which is related to the number of required colors that the two algorithms used in finding a solution. When increasing the density of the graph, for example for values greater than 85%, the number of required colors increases significantly, and this is true for both algorithms - GC-IDX algorithm and GC-RND algorithm. However, the number of colors needed to color the vertices of a graph, even at a density of 95% does not exceed more than half the number of vertices in the same graph, since 394/1000=0.394. This is an important result of how increasing the density of the graph affects the number of chromatic classes generated by the algorithms. It is known that for a complete graph, *i.e.*, at 100% density, the number of colors required to color a graph equal to the number of vertices of that graph. This means that when the density of the graph changes between 95% and 100%, the number of colors needed to color the graph increases significantly. This is exactly what the results presented in Table 2. It can also be seen that at high values of the graph density parameter, both algorithms generate identical solutions, such as the results obtained for graphs with density above 97%.

The chart in Figure 6 illustrates how increasing the number of edges (and thus the graph density) impacts the execution time of the two algorithms. The execution time of the GC-RND algorithm is significantly longer than that of the GC-IDX algorithm. This difference varies: for example, in graph G1, which has the lowest density, the difference is the largest, while in graph G19, which has the highest density, the difference is the smallest. However, for each run of the GC-RND algorithm, 100 iterations are performed, and the iteration that generates the best solution is selected. The times presented in Table 1 include the execution time of all 100 iterations. This means that the actual time for a single execution of the algorithm is, on average, 100 times less. The complexity of both algorithms is quadratic, primarily depending on the number of vertices in the graph, and to a lesser extent on its density, *i.e.*, the number of edges. This is illustrated in Figure 6, where a significant increase in the number of edges, and thus the graph's density, results in the execution time of both algorithms changing insignificantly and linearly, with only a very small increment.

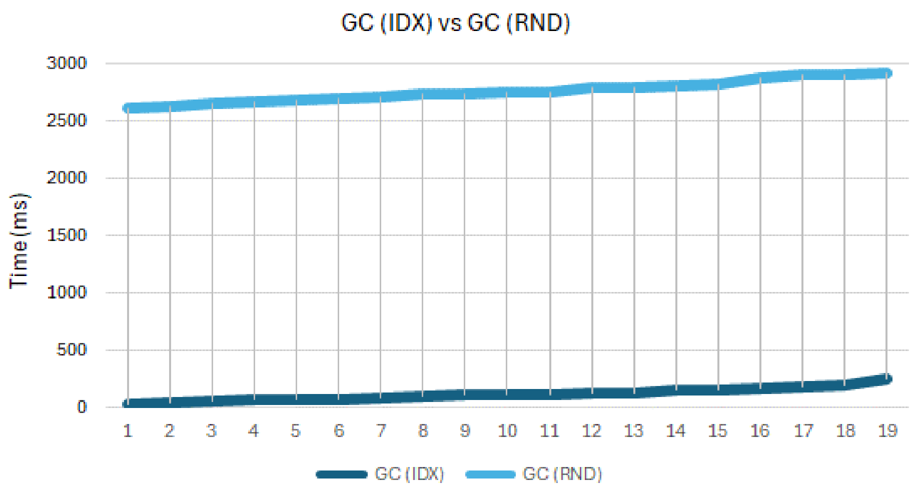

Figure 6. Comparison of the execution times of both algorithms

## 4. CONCLUSION

This paper discusses a study related to the graph labeling (coloring) problem. Various scientific analyses, methods, and algorithms related to the graph labeling (coloring) problem were discussed. In this paper, two approximate algorithms for solving the graph vertex coloring problem were implemented and presented. The declarations of the basic data structures used by the algorithms, including one-dimensional and two-dimensional arrays, were also presented. The program codes of both heuristic algorithms were presented and analyzed in detail. When conducting the experiments, the operating system's ability to work in multitasking mode was specifically considered. Accordingly, six runs of both algorithms were conducted for each of the 24 analyzed graphs. The average execution times from these runs were calculated and presented in Tables 1 and 2. For the solutions generated by the GC-IDX algorithm, identical results were obtained in all runs. These results are also presented in Table 1 and Table 2. In contrast, the GC-RND algorithm produced different results across the six runs. Table 1 and Table 2 present the best results generated for each graph across all runs.

The results of this experiment show that, for graphs G1 – G19, the GC-IDX algorithm generated better solutions than those generated by the GC-RND algorithm in most cases. In only two cases did the GC-RND algorithm generate solutions that were comparable to those produced by the GC-IDX algorithm. The results indicate another trend: as the graph density exceeds 85%, the number of required colors increases significantly for both algorithms. However, even at a density of 95%, the number of colors required to color the vertices of a graph does not exceed half the number of vertices in the graph. This is an important finding that demonstrates how increasing the graph's density affects the number of chromatic classes generated by the algorithms. As the graph density increases from 95% to 100%, the number of colors required to color the graph rises significantly. However, for graph density values above 97%, both algorithms generate identical solutions. The complexity of both algorithms is quadratic, primarily depending on the number of vertices in the graph and, to a lesser extent, on its density, *i.e.*, the number of edges. The results indicate that with a significant increase in graph density, *i.e.*, a substantial increase in the number of edges, the execution time of both algorithms changes insignificantly and almost linearly, with only a very small increment.

## FUNDING INFORMATION

This research was conducted without financial support from external sources.

## AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

| Name of Author | C | M | So | Va | Fo | I | R | D | O | E | Vi | Su | P | Fu |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Velin Kralev | ✓ | ✓ |  | ✓ | ✓ | ✓ |  |  | ✓ | ✓ |  | ✓ | ✓ | ✓ |
| Radoslava Kraleva | ✓ | ✓ | ✓ | ✓ |  | ✓ |  | ✓ | ✓ | ✓ | ✓ |  | ✓ | ✓ |

|  |  |  |
|---|---|---|
| C : **C**onceptualization | I : **I**nvestigation | Vi : **Vi**sualization |
| M : **M**ethodology | R : **R**esources | Su : **Su**pervision |
| So : **So**ftware | D : **D**ata Curation | P : **P**roject administration |
| Va : **Va**lidation | O : Writing - **O**riginal Draft | Fu : **Fu**nding acquisition |
| Fo : **Fo**rmal analysis | E : Writing - Review & **E**diting |  |

## CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

## DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author, VK, upon reasonable request.

## REFERENCES

[1] S. Slamin, N. O. Adiwijaya, M. A. Hasan, D. Dafik, and K. Wijaya, "Local super antimagic total labeling for vertex coloring of graphs," *Symmetry*, vol. 12, no. 11, p. 1843, Nov. 2020, doi: 10.3390/sym12111843.

[2] T. Emden-Weinert, S. Hougardy, and B. Kreuter, "Uniquely colourable graphs and the hardness of colouring graphs of large girth," *Combinatorics, Probability and Computing*, vol. 7, no. 4, pp. 375–386, Dec. 1998, doi: 10.1017/S0963548398003678.

[3] A. Punitha and G. Jayaraman, "Computation of total chromatic number for certain convex polytope graphs," *Journal of Applied Mathematics and Informatics*, vol. 42, no. 3, pp. 567–582, 2024, doi: 10.14317/jami.2024.567.

[4] S. Nicoloso and U. Pietropaoli, "Vertex-colouring of 3-chromatic circulant graphs," *Discrete Applied Mathematics*, vol. 229, pp. 121–138, Oct. 2017, doi: 10.1016/j.dam.2017.05.013.

[5] M. R. Garey, D. S. Johnson, and L. Stockmeyer, "Some simplified NP-complete graph problems," *Theoretical Computer Science*, vol. 1, no. 3, pp. 237–267, Feb. 1976, doi: 10.1016/0304-3975(76)90059-1.

[6] F. Lehner and S. M. Smith, "On symmetries of edge and vertex colourings of graphs," *Discrete Mathematics*, vol. 343, no. 9, p. 111959, Sep. 2020, doi: 10.1016/j.disc.2020.111959.

[7] D. S. Malyshev and O. I. Duginov, "A complete complexity dichotomy of the edge-coloring problem for all sets of -edge forbidden subgraphs," *Journal of Applied and Industrial Mathematics*, vol. 17, no. 4, pp. 791–801, Sep. 2023, doi: 10.1134/S1990478923040099.

[8] V. Kralev and R. Kraleva, "An analysis between different algorithms for the graph vertex coloring problem," *International Journal of Electrical and Computer Engineering*, vol. 13, no. 3, pp. 2972–2980, 2023, doi: 10.11591/ijece.v13i3.pp2972-2980.

[9] S. Ghosal and S. C. Ghosh, "Expected polynomial-time randomized algorithm for graph coloring problem," *Discrete Applied Mathematics*, vol. 354, pp. 108–121, 2024, doi: 10.1016/j.dam.2023.08.001.

[10] C. López-Ramírez, J. E. Gutiérrez Gómez, and G. De Ita Luna, "Building a maximal independent set for the vertex-coloring problem on planar graphs," *Electronic Notes in Theoretical Computer Science*, vol. 354, pp. 75–89, 2020, doi: 10.1016/j.entcs.2020.10.007.

[11] Z. Huanping, Z. Peijin, L. Jingwen, and S. Huojie, "Novel algorithm for adjacent vertex-distinguishing edge coloring of large-scale random graphs," *Journal of Engineering Science and Technology Review*, vol. 14, no. 3, pp. 69–75, 2021, doi: 10.25103/jestr.143.08.

[12] P. T. Lima, E. J. van Leeuwen, and M. van der Wegen, "Algorithms for the rainbow vertex coloring problem on graph classes," *Theoretical Computer Science*, vol. 887, pp. 122–142, Oct. 2021, doi: 10.1016/j.tcs.2021.07.009.

[13] K. Kanahara, K. Katayama, and E. Tomita, "Speeding-up construction algorithms for the graph coloring problem," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E105.A, no. 9, p. 2021DMP0011, Sep. 2022, doi: 10.1587/transfun.2021DMP0011.

[14] R. Martín-Santamaría, M. López-Ibáñez, T. Stützle, and J. M. Colmenar, "On the automatic generation of metaheuristic algorithms for combinatorial optimization problems," *European Journal of Operational Research*, vol. 318, no. 3, pp. 740–751, 2024, doi: 10.1016/j.ejor.2024.06.001.

[15] K. H. Alnafisah, "Enhancing algorithmic techniques for streamlined complex graph structures in big data visualization," *Engineering, Technology and Applied Science Research*, vol. 15, no. 2, pp. 21159–21165, 2025, doi: 10.48084/etasr.9740.

[16] U. Fatima, S. Hina, and M. Wasif, "Analysis of community groups in large dynamic social network graphs through fuzzy computation," *Systems and Soft Computing*, vol. 7, 2025, doi: 10.1016/j.sasc.2025.200239.

[17] T. Karthick, F. Maffray, and L. Pastor, "Polynomial cases for the vertex coloring problem," *Algorithmica*, vol. 81, no. 3, pp. 1053–1074, Mar. 2019, doi: 10.1007/s00453-018-0457-y.

[18] M. Chudnovsky, T. Karthick, P. Maceli, and F. Maffray, "Coloring graphs with no induced five-vertex path or gem," *Journal of Graph Theory*, vol. 95, no. 4, pp. 527–542, 2020, doi: 10.1002/jgt.22572.
[19] M. Zaker, "A new vertex coloring heuristic and corresponding chromatic number," *Algorithmica*, vol. 82, no. 9, pp. 2395–2414, Sep. 2020, doi: 10.1007/s00453-020-00689-4.
[20] D. Goyal and R. Jaiswal, "Tight FPT approximation for constrained k-center and k-supplier," *Theoretical Computer Science*, vol. 940, pp. 190–208, Jan. 2023, doi: 10.1016/j.tcs.2022.11.001.
[21] S. Fujita, S. Kitaev, S. Sato, and L.-D. Tong, "On properly ordered coloring of vertices in a vertex-weighted graph," *Order*, vol. 38, no. 3, pp. 515–525, Oct. 2021, doi: 10.1007/s11083-021-09554-7.
[22] Y. Uchida, K. Yajima, and K. Haraguchi, "Recycling solutions for vertex coloring heuristics," *Journal of the Operations Research Society of Japan*, vol. 64, no. 3, pp. 184–202, 2021, doi: 10.15807/jorsj.64.184.
[23] K. Oshiro and N. Oyamaguchi, "Palettes of Dehn colorings for spatial graphs and the classification of vertex conditions," *Journal of Knot Theory and Its Ramifications*, vol. 30, no. 03, p. 2150015, Mar. 2021, doi: 10.1142/S0218216521500152.
[24] J. Mangaiyarkkarasi, J. S. Revathy, and S. Mehta, "Introduction to graph theory," in *Neural Networks and Graph Models for Traffic and Energy Systems*, IGI Global, 2025, pp. 65–82.
[25] M. Jonckheere and M. Sáenz, "Asymptotic optimality of degree-greedy discovering of independent sets in configuration model graphs," *Stochastic Processes and their Applications*, vol. 131, pp. 122–150, 2021, doi: 10.1016/j.spa.2020.09.009.
[26] F. Bonomo-Braberman *et al.*, "Better 3-coloring algorithms: Excluding a triangle and a seven vertex path," *Theoretical Computer Science*, vol. 850, pp. 98–115, Jan. 2021, doi: 10.1016/j.tcs.2020.10.032.
[27] G. S. Terci and B. Boz, "Coloring dynamic graphs with a similarity and pool-based evolutionary algorithm," *IEEE Access*, vol. 13, pp. 38054–38075, 2025, doi: 10.1109/ACCESS.2025.3546108.

## BIOGRAPHIES OF AUTHORS

**Velin Kralev** is an associate professor of computer science at the Faculty of Mathematics and Natural Sciences, South-West University, Blagoevgrad, Bulgaria. He defended his Ph.D. thesis in 2010. His research interests include database systems development, optimization problems of the scheduling theory, graph theory, and component-oriented software engineering. He can be contacted at email: velin_kralev@swu.bg.

**Radoslava Kraleva** is an associate professor of computer science at the Faculty of Mathematics and Natural Sciences, South-West University "Neofit Rilski", Blagoevgrad, Bulgaria. She defended her Ph.D. thesis "Acoustic-Phonetic Modeling for Children's Speech Recognition in Bulgarian" in 2014. Her research interests include child-computer interaction, speech recognition, mobile app development and computer graphic. She is an editorial board member and reviewer of many journals. She can be contacted at email: rady_kraleva@swu.bg.