International Journal of Electrical and Computer Engineering (IJECE)
Vol. 13, No. 3, June 2023, pp. 2981~2989
ISSN: 2088-8708, DOI: 10.11591/ijece.v13i3.pp2981-2989

A comparative analysis between two heuristic 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 May 24, 2022
Revised Sep 5, 2022
Accepted Oct 1, 2022

Keywords:

Graph coloring
Graph theory
Greedy coloring
Heuristic algorithm
Sequential coloring

ABSTRACT

This study focuses on two heuristic algorithms for the graph vertex coloring problem: the sequential (greedy) coloring algorithm (SCA) and the Welsh–Powell algorithm (WPA). The code of the algorithms is presented and discussed. The methodology and conditions of the experiments are presented. The execution time of the algorithms was calculated as the average of four different starts of the algorithms for all analyzed graphs, taking into consideration the multitasking mode of the operating system. In the graphs with less than 600 vertices, in 90% of cases, both algorithms generated the same solutions. In only 10% of cases, the WPA algorithm generates better solutions. However, in the graphs with more than 1,000 vertices, in 35% of cases, the WPA algorithm generates better solutions. The results show that the difference in the execution time of the algorithms for all graphs is acceptable, but the quality of the solutions generated by the WPA algorithm in more than 20% of cases is better compared to the SC algorithm. The results also show that the quality of the solutions is not related to the number of iterations performed by the algorithms.

This is an open access article under the CC BY-SA license.

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

A graph vertex coloring is an assignment of a certain color on each of the vertices of a given graph. Each color is exactly one element of a predefined set of colors, such as *S*. The vertices of a graph that are colored the same color form a color class. If there are exactly *k* elements in the set *S*, then the coloring of the vertices of the graph is called *k*-coloring. Integers from 1 to *k* are usually used to denote elements in *S* (i.e., colors). It is assumed that one coloring is proper if every two adjacent vertices in a graph are colored differently. The following formulation can be made that a graph is *k*-colorable if it has *k*-coloring that is proper. It is easily established that such a coloring exists if the set *S* is initialized with |V| number of elements, i.e., as is the number of vertices in the graph *G*. In this case, if a different color is assigned to each vertex corresponding to exactly one element of *S*, then an acceptable (proper) coloring will certainly be obtained. In this coloring, there will certainly not be two vertices that are colored the same color [1], [2].

The optimal coloring of a graph *G* is denoted by $x(G)$. If graph *G* is not a complete graph, then $x(G)$ is less than |V|. If for graph *G* it is found that $x(G) = k$, then for this graph it can be said that it is *k*-chromatic. Each color class is stable if the coloring of a graph is proper. Thus (proper) coloring a graph with *k* number of colors actually represents the grouping of the vertices of this graph in *k* number of disjoint sets. When a graph is *k*-colorable, it is called a *k*-partite graph. That is why a 2-colorable graph is also called a bipartite graph. If two graphs *G* and *G*' are given, and if the graph *G*' is a subgraph of graph *G*, then each proper coloring of *G* is the proper coloring of G'. In addition, the chromatic number of graph G' is less than or equal to the chromatic number of graph G [3], [4].

The vertex coloring problem (in graph theory) is an NP-hard problem [5] and it is still being examined [6], [7]. Different aspects of this problem are discussed in scientific literature. For instance, the rainbow vertex coloring problem [8], the adjacent vertex-distinguishing edge coloring [9], and the maximal independent set for the vertex-coloring problem on planar graphs [10]. Distinct aspects of the problem use various techniques [11]–[13], algorithms [14], [15] and approaches [16], [17]. Other algorithms and approaches are used to solve similar problems in graphs [18], [19]. Other in-depth analyses of this problem are presented in [20]–[22].

A complete graph K_n can be colored with exactly n number of colors, because each vertex is adjacent to all other vertices, i.e., for a complete K_n, the chromatic number coincides with the number of vertices n, i.e., $\chi(K_n)=n$. From this it can also be concluded that if in a graph G there exists a complete subgraph of it, then the chromatic number of G (i.e., $x(G)$) will be a value greater than or at least equal to the number of vertices forming the complete subgraph (clique number) of G, i.e., $x(G) \geq \omega(G)$. It has also been found that a graph can have a larger chromatic number than the power of the set of vertices forming a complete subgraph of a given graph [23], [24].

Some bounds on the chromatic number have been obtained in the development of algorithms for coloring the vertices of a graph. A commonly used greedy algorithm for the approximate coloring of graph vertices is based on the use of vertex degrees [25]. The vertices are colored sequentially in descending order of their degrees. In this situation, the coloring of any vertex will not require more colors than the degree of the vertex and another color for the vertex itself. This is because even if all vertices adjacent to a given vertex are colored differently (i.e., a different color is used for each adjacent vertex), in the worst case it will be necessary at most more than one color for the current vertex. As a consequence of the development and use of this greedy algorithm for graph vertex coloring, the bound $x(G) \leq \Delta(G)+1$ is obtained, where $\Delta(G)$ is the largest degree of a vertex in G [26], [27].

The coloring of a graph with $\Delta(G)+1$ colors is the worst possible result that can be generated by the greedy algorithm. If the same algorithm is used, but the order of the vertices is different, then the generated result may be better than the last one found. There can be no worse result than $\Delta(G)+1$. However, finding the "right" ordinance to generate the optimal solution requires $|V|!$ checks. This is due to the fact that finding the chromatic number of a graph is an NP-hard problem [5].

In this paper, two heuristic algorithms for the graph vertex coloring problem will be presented and analyzed-the sequential coloring algorithm (SCA) [25] and the Welsh–Powell algorithm (WPA) [28]. These algorithms are approximate and they do not always find optimal solutions. This type of algorithm is used when the problem is NP-hard and when the input data is large (in terms of the number of vertices and edges in a graph). In addition, there are other algorithms for the graph vertex coloring problem [29], [30].

2. RESEARCH METHOD

This section presents detailed implementations of the SCA and WPA algorithms. Both algorithms are heuristic and are used to approximately solve the graph vertex coloring problem. For both algorithms, some global variables and data structures need to be declared in advance. They are shown in Figure 1.

```
01 var
02   VertexCount: Integer;
03   VertexColor: array of TColor;
04   MinColors: Integer;
05   AdjMatrix: array of array of Integer;
06   Counter, MiddleCounter, ExternalCounter: Int64;
```

Figure 1. Code of the global declarations

The *VertexCount* variable (line 2) stores the number of vertices in the graph. The *VertexColor* dynamic array of type *TColor*, which is declared on line 3, is used by both algorithms. Each element of this array contains a color with which the corresponding vertex of the graph is colored. Coloring algorithms change the values of these elements. The *MinColors* variable (declared on line 4) is aggregate and is used by coloring algorithms in the solution search process. Each graph is represented by an adjacency matrix, which is declared on line 5. Each element [i, j] of the matrix indicates whether the vertices with indices i and j are adjacent or not.

The sequential coloring method implements the first heuristic algorithm for coloring graph vertices. The code of this algorithm is presented in Figure 2. It uses additional (local) variables: Color, Index, and

IsFeasible. The Color variable contains the index of one of the colors used so far. The variable index is used when searching for the adjacent of the current vertex. The *IsFeasible* variable (of type Boolean) indicates whether the current vertex can be colored with one of the available colors or not.

```
01 procedure SequentialColoring;
02 var
03   IsFeasible: Boolean;
04   Col, Index, Color: Integer;
05 begin
06   Counter := 0;
07   MinColors := 0;
08   MiddleCounter := 0;
09   ExternalCounter := 0;
10   for Index := 1 to VertexCount do
11   begin
12     ExternalCounter := ExternalCounter + 1;
13     Color := 0;
14     repeat
15       MiddleCounter := MiddleCounter + 1;
16       Color := Color + 1;
17       IsFeasible := True;
18       for Col := 1 to VertexCount do
19       begin
20         Counter := Counter + 1;
21         if ((AdjMatrix[Index][Col] > 0) and
22             (VertexColor[Col] = Color)) then
23         begin
24           IsFeasible := False;
25           Break;
26         end;
27       end;
28     until (IsFeasible = True);
29     VertexColor[Index] := Color;
30     if (MinColors < Color) then MinColors := Color;
31   end;
32 end;
```

Figure 2. Code of the sequential coloring algorithm

The global variables *MinColors, ExternalCounter*, *MiddleCounter*, and *Counter* are initialized to 0 in the body of the sequential coloring method (lines 6-9). The traversal of the vertices is realized by a for-loop (line 10). The algorithm checks which of the available colors can be used to color the current vertex. The color to be chosen should be as small as possible. This check is realized by a repeat loop (lines 14-28). Immediately before the loop, the local variable *Color* is initialized to 0 (line 13). At the beginning of the loop, the value of the local variable *Color* is incremented by 1 (line 16), which in the first iteration means that this variable will be set to 1. The current vertex can be colored with the current color only if none of its adjacents is colored with this color. This check is performed via the for-loop (lines 18-27). For each adjacent vertex of the current vertex, check that it is not colored with a color stored as a number in the *Color* variable. If a vertex colored with this color is found, the loop is immediately interrupted (line 25), but the local variable *IsFeasible* is first set to False (line 24). This means that the current vertex cannot be colored with the current color. Since the value of the local variable *IsFeasible* is False, the end of loop condition (line 28) will not be met and therefore a new iteration will be performed. When the new iteration starts, the color number is increased by one (line 16). In this way, the algorithm starts checking whether the current vertex can be colored with a color number *Color+1*. The repeat loop (lines 14-28) ends only when a suitable color is found for the current vertex. In this situation, the variable *IsFeasible* will be equal to *True* (set on line 17 at the beginning of the current iteration). The current color (the value of the variable *Color*) will be stored in the dynamic array *VertexColor* in the element indicated by the variable Index, i.e. the current vertex (line 29). If the value of the local variable *Color* is greater than the value of the global variable MinColors, then this value is also set to the global variable *MinColors* (line 30). This means that a new color has been added to the existing ones because coloring the current vertex with one of the available colors was not possible. Once all the vertices of the graph are colored, i.e. the execution of the external for-loop (lines 10-31) is completed, the minimum number of required colors and the number of iterations performed by the three nested loops will be stored in the variables *MinColors*, *ExternalCounter*, *MiddleCounter*, and *Counter*.

The Welsh–Powell method implements the second heuristic algorithm for coloring graph vertices. The code of this algorithm is presented in Figure 3. Local variables: *Col, Index, Color*, and *IsFeasible* have the same meaning as those declared in the sequential coloring method.

```
01 procedure WelshPowell;
02 var
03   IsFeasible: Boolean;
04   Col, Index, Color, ColoredVertices: Integer;
05 begin
06   Color := 0; MinColors := 0; Counter := 0;
07   MiddleCounter := 0; ExternalCounter := 0; ColoredVertices := 0;
08   while (not (ColoredVertices = VertexCount)) do
09   begin
10     ExternalCounter := ExternalCounter + 1; Color := Color + 1;
11     for Index := 1 to VertexCount do
12     begin
13       MiddleCounter := MiddleCounter + 1;
14       if (VertexColor[Index] = 0) then
15       begin
16         IsFeasible := True;
17         for Col := 1 to VertexCount do
18         begin
19           Counter := Counter + 1;
20           if ((AdjMatrix[Index][Col] > 0) and
21               (VertexColor[Col] = Color)) then
22             begin IsFeasible := False; Break; end;
23         end;
24         if (IsFeasible = True) then
25         begin
26           VertexColor[Index] := Color;
27           if (MinColors < Color) then MinColors := Color;
28           ColoredVertices := ColoredVertices + 1;
29 end; end; end; end; end;
```

Figure 3. Code of the WPA

The local variable *ColoredVertices* (of type *Integer*) shows the current number of colored vertices in the graph. In the body of the Welsh–Powell method, the global variables *MinColors*, *ExternalCounter*, *MiddleCounter*, and *Counter*, as well as the local variables *Color* and *ColoredVertices* (lines 6-7) are set to 0. The process of coloring vertices is performed until all vertices of the graph are colored (line 8-the condition for the end of the while loop). Once the next color is selected (line 10) the algorithm traverses all vertices of the graph (through a for loop, which starts at line 11). Then, only for uncolored vertices, the algorithm checks whether any vertex adjacent to the current vertex is not colored with the current color (line 14). This check is done through the loop implemented between lines 17-23. This “for” loop iterates through all vertices and checks those of them that are adjacent to the current vertex. Once all adjacent vertices of the current vertex are checked and there is no one that is colored with the current color (the value of the local variable *Color*), the value of the Boolean variable *IsFeasible* will not be changed and will be equal to True. In this situation, the current vertex will be assigned the current color (line 26). The code of line 27 checks whether the value of the local variable *Color* is greater than the value of the global variable *MinColor*. If this is true, then the number of colors used is greater than the last registered one and the value of the global variable *MinColors* will be updated (line 27). The computational complexity of both heuristic algorithms (SCA and WPA) is quadratic and depends on the number of vertices of the graph-*VertexCount*).

3. RESULTS AND DISCUSSION

The results of the experiment will be shown and discussed. A comparative analysis between heuristic algorithms, in terms of the quality of the generated solutions and the time for their finding, will be presented and analyzed as well. For this research, 40 graphs, respectively with 30, ..., and 20000 vertices were created. These graphs were divided into two sets, the first one contained 20 graphs, and the second one the remaining 20 graphs. In this distribution, the first set included the graphs with 30÷600 vertices, and the second set, the graphs with 1000÷20000 vertices. These graphs are presented in Tables 1 and 2. Up to 20% of the possible edges are used in each graph. The experimental conditions are 32-bit Win 10 OS and hardware configuration: Processor: Intel (R) Core (TM) i5-1135G7 at 2.40-4.20 GHz; RAM: 16GB DDR4. Both sets of graphs are used to conduct experiments with both heuristic algorithms. All graphs are generated randomly, and for each graph, the specific information is shown in Tables 1 and 2.

In Tables 3 and 4, the “External”, “Middle”, and “Internal” columns show the number of iterations that the algorithms have made to find the solutions for all graphs. These solutions show the number of different colors needed to color the vertices of the graphs and arrange these vertices into chromatic classes. These values are shown in the Colors columns below the SC and WP columns in Table 3, and below the SCA and the WPA columns in Table 4. The execution time of both algorithms for the graphs of the first set is very short and therefore it is not presented. The times of both algorithms for the second set of graphs are shown in the “Time (ms)” columns in Table 4.

Table 1. The first set of graphs

Graph abbr.	Vertex count	Edge count	Vertices degree			Graph caption	Vertex count	Edge count	Vertices degree		
			Min	Max	Avg				Min	Max	Avg
G01	30	87	2	12	6	G11	330	10 857	46	86	66
G02	60	354	4	20	12	G12	360	12 924	50	94	72
G03	90	801	10	25	18	G13	390	15 171	56	100	78
G04	120	1 428	11	35	24	G14	420	17 598	63	112	84
G05	150	2 235	16	42	30	G15	450	20 205	63	116	90
G06	180	3 222	22	49	36	G16	480	22 992	74	121	96
G07	210	4 389	27	55	42	G17	510	25 959	74	135	102
G08	240	5 736	33	66	48	G18	540	29 106	80	132	108
G09	270	7 263	36	71	54	G19	570	32 433	87	143	114
G10	300	8 970	44	80	60	G20	600	35 940	94	153	120

Table 2. The second set of graphs

Graph abbr.	Vertex count	Edge count	Vertices degree			Graph caption	Vertex count	Edge count	Vertices degree		
			Min	Max	Avg				Min	Max	Avg
G21	1 000	99 900	152	237	200	G31	11 000	12 098 900	2 040	2 361	2 200
G22	2 000	399 800	332	464	400	G32	12 000	14 398 800	2 233	2 549	2 400
G23	3 000	899 700	524	674	600	G33	13 000	16 898 700	2 427	2 778	2 600
G24	4 000	1 599 600	710	895	800	G34	14 000	19 598 600	2 599	3 012	2 800
G25	5 000	2 499 500	897	1 098	1 000	G35	15 000	22 498 500	2 816	3 212	3 000
G26	6 000	3 599 400	1 088	1 326	1 200	G36	16 000	25 598 400	3 018	3 382	3 200
G27	7 000	4 899 300	1 261	1 518	1 400	G37	17 000	28 898 300	3 184	3 603	3 400
G28	8 000	6 399 200	1 465	1 747	1 600	G38	18 000	32 398 200	3 376	3 841	3 600
G29	9 000	8 099 100	1 646	1 964	1 800	G39	19 000	36 098 100	3 583	4 029	3 800
G30	10 000	9 999 000	1 871	2 158	2 000	G40	20 000	39 998 000	3 780	4 248	4 000

Table 3. Results of the heuristic algorithms for the first set of graphs

Graph abbr.	Colors		External		Middle		Internal		Graph abbr.	Colors		External		Middle		Internal	
	SC	WP	SC	WP	SC	WP	SC	WP		SC	WP	SC	WP	SC	WP	SC	WP
G01	5	5	30	5	82	150	1 191	1 191	G11	23	23	330	23	3 357	7 590	300 091	300 091
G02	7	7	60	7	220	420	5 212	5 212	G12	24	23	360	23	3 841	8 280	366 272	369 394
G03	9	9	90	9	398	810	13 596	13 596	G13	25	25	390	25	4 557	9 750	475 789	475 789
G04	12	12	120	12	633	440	26 230	26 230	G14	27	27	420	27	5 170	11 340	566 661	566 661
G05	13	13	150	13	917	1 950	45 185	45 185	G15	28	28	450	28	5 827	12 600	676 423	676 423
G06	14	14	180	14	1 183	2 520	65 695	65 695	G16	28	28	480	28	6 276	13 440	752 864	752 864
G07	16	16	210	16	1 533	3 360	98 083	98 083	G17	31	31	510	31	7 249	15 810	953 025	953 025
G08	17	17	240	17	1 922	4 080	134 376	134 376	G18	33	32	540	32	7 825	17 280	1 083 227	1 062 262
G09	20	20	270	20	2 360	5 400	176 384	176 384	G19	34	34	570	34	8 640	19 380	1 265 291	1 265 295
G10	20	20	300	20	2 733	6 000	229 929	229 929	G20	35	35	600	35	9 394	21 000	1 432 981	1 432 981

Table 4. Results of the heuristic algorithms for the second set of graphs

Graph abbr.	Sequential coloring algorithm					Welsh–Powell algorithm				
	Colors	External	Middle	Internal	Time (ms)	Colors	External	Middle	Internal	Time (ms)
G21	52	1 000	23 543	5 695 311	47	52	52	52 000	5 695 311	47
G22	89	2 000	80 727	39 605 006	375	89	89	178 000	39 605 207	672
G23	123	3 000	168 091	124 699 884	1 438	123	123	369 000	124 700 070	2 671
G24	154	4 000	284 184	278 084 547	3 922	154	154	616 000	278 086 086	6 719
G25	189	5 000	432 340	539 491 573	10 859	189	189	945 000	539 493 041	14 609
G26	219	6 000	601 324	908 936 870	20 172	218	218	1 308 000	906 350 187	26 579
G27	249	7 000	805 175	1 431 247 938	34 078	249	249	1 743 000	1 431 249 009	43 141
G28	279	8 000	1 025 821	2 077 723 571	52 234	278	278	2 224 000	2 080 168 718	65 531
G29	307	9 000	1 273 997	2 923 844 144	76 578	306	306	2 754 000	2 911 093 949	93 703
G30	334	10 000	1 540 075	3 938 157 114	110 750	334	334	3 340 000	3 938 161 248	127 828
G31	363	11 000	1 843 987	5 204 754 435	144 141	363	363	3 993 000	5 204 760 013	173 437
G32	394	12 000	2 176 084	6 758 608 019	192 453	392	392	4 704 000	6 748 580 090	235 563
G33	419	13 000	2 520 491	8 459 404 351	261 156	419	419	5 447 000	8 459 412 267	290 937
G34	448	14 000	2 901 825	10 582 260 636	315 891	446	446	6 244 000	10 568 032 443	365 625
G35	474	15 000	3 302 805	12 935 704 083	378 203	474	474	7 110 000	12 929 018 308	442 796
G36	503	16 000	3 728 281	15 623 115 566	473 375	503	503	8 048 000	15 623 125 926	542 063
G37	526	17 000	4 154 223	18 391 259 590	571 781	526	526	8 942 000	18 391 268 176	645 859
G38	554	18 000	4 620 350	21 770 668 415	698 156	554	554	9 972 000	21 826 875 897	780 984
G39	582	19 000	5 125 921	25 720 545 690	833 906	582	582	11 058 000	25 699 056 417	942 562
G40	607	20 000	5 638 245	29 716 918 660	917 922	606	606	12 120 000	29 715 576 882	1 073 500

Tables 3 and 4 and the charts in Figures 4 and 5 show the results of the algorithms for the number of chromatic classes (colors) generated for the two sets of graphs (G01-G20 and G21-G40). The results also show that in the first set of graphs (G01-G20) only in two cases (G12 and G18) the WP algorithm has found better solutions compared to the SC algorithm. In these two cases, the number of internal iterations performed by the algorithms is different. For all other cases, both algorithms generated the same solutions. For graph G12, the WPA algorithm performed 3 122 more iterations than the SCA algorithm. For graph G18, the opposite is true. Although the WPA algorithm has generated a better solution for this graph, the iterations performed by it are 20 965 less than those performed by the SCA algorithm. The results for the graphs of set 1 show that the WPA algorithm generates in some cases better solutions than the SCA algorithm, but the quality of these solutions is not necessarily related to a greater number of iterations performed by the WPA algorithm. In addition, even with a different number of internal iterations performed by the algorithms, the generated solutions may be equal, as in graph G19.

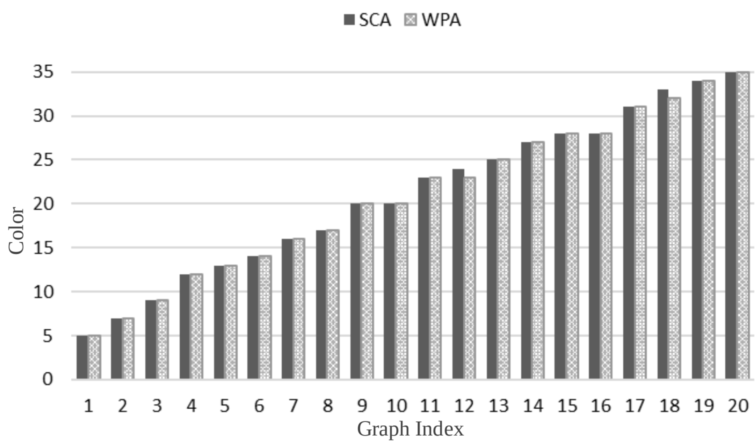

Figure 4. The number of colors generated from the algorithms for the graphs of set 1

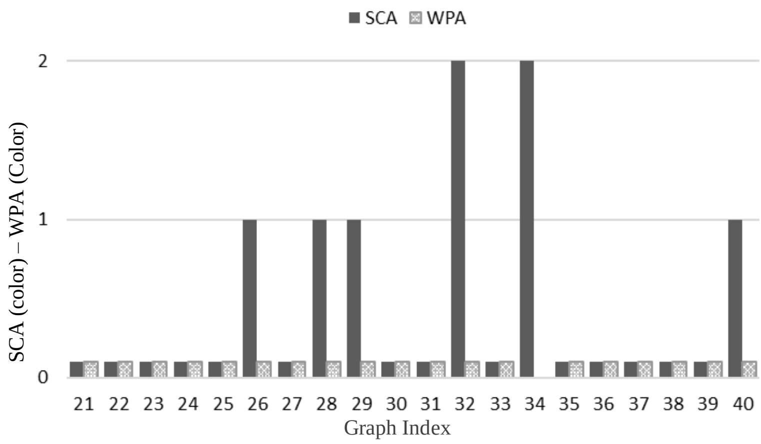

Figure 5. Difference between the number of colors generated from both algorithms for the graphs of set 2

The results of the second set of graphs (G20-G40) show that in six cases (G26, G28, G29, G32, G34, and G40) the WPA algorithm has found better solutions compared to the SCA algorithm. Table 4 and the chart in Figure 5 show that in 14 cases both algorithms generated identical solutions. In 4 cases (G26, G28, G29, and G40) the WPA algorithm found solutions differing by only one color from those generated by the SCA algorithm. However, in 2 cases (G32 and G34) the WPA algorithm found solutions differing by two colors from those generated by the SCA algorithm. This improvement is significant for the graph vertex coloring problem in cases where the generated solutions are close to the optimal solution.

The chart in Figure 6 shows the differences between the internal iterations of the two algorithms (for all graphs in set 2). Although there is no direct relationship between the number of these iterations and the quality of the solutions generated by the algorithms, it can be noted that in 5 out of 6 cases of different solutions (graphs G26, G29, G32, G34, and G40) the number of performed internal iterations of the SCA algorithm is significantly larger than those performed of the WPA algorithm. The generated solutions by the SC algorithm are worse in these graphs, which is not the case only in graph G28.

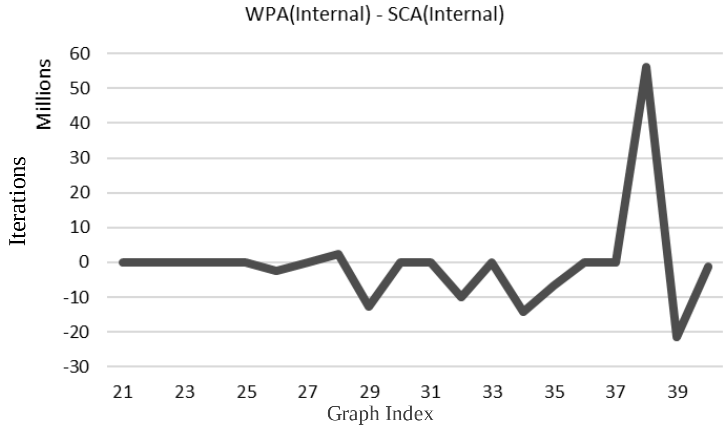

Figure 6. Difference between the number of internal iterations generated from two algorithms

The chart in Figure 7 shows the effect of increasing the size of the graphs (increasing the number of vertices and edges) on the execution time of both algorithms. The execution time of the WPA algorithm is longer than that of the SCA algorithm, but the difference is in minutes. For example, for graph G39, the execution time of the SCA algorithm is 13 minutes and 54 seconds, and the execution time of the WPA algorithm for the same graph is 15 minutes and 43 seconds. The difference between the times is 1 minute and 49 seconds. For graph G40, the execution time of the SCA algorithm is 15 minutes and 18 seconds, and the execution time of the WPA algorithm for the same graph is 17 minutes and 54 seconds. The difference between the times is 2 minutes and 36 seconds.

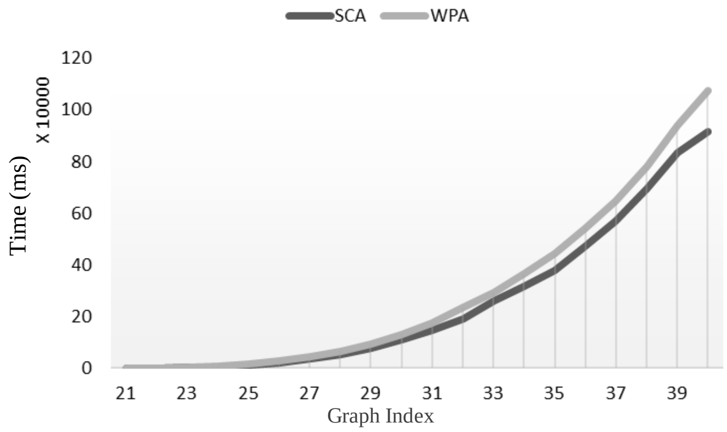

Figure 7. Comparison of the execution times of both algorithms for the graphs of set 2

4. CONCLUSION

In this paper, a study related to the graph coloring problem was presented. Various scientific publications discussing this problem and related different approaches and methods for solving it were also presented. Two heuristic algorithms for solving the problem: the Sequential coloring algorithm (SCA) and the WPA were implemented and analyzed. The global declarations of data structures used by the algorithms (variables, arrays, and matrices) were shown. The source code of the heuristic algorithms was implemented, presented, and analyzed in detail. Taking into account the multitasking mode of the operating system, the execution time of the algorithms was calculated as the average of four different starts of the two algorithms for each of the 40 analyzed graphs (of the two sets).

The results show that the WPA algorithm generates in some cases better solutions than the SCA algorithm, but the quality of these solutions is not necessarily related to a greater number of iterations performed by the WPA algorithm. In the first set of graphs, in 18 out of 20 cases, both algorithms generated the same solutions. In only 2 of these 20 cases, the WPA algorithm generates better solutions compared to the SCA algorithm. In the second set of graphs, in 13 out of 20 cases, both algorithms generated the same solutions, but in the remaining 7 cases, the WPA algorithm generated better solutions compared to the SCA algorithm. In addition, in 2 of these 7 cases, the improvement was two chromatic classes less than one, as in the other 5 cases. In summary, for the second set of graphs the WPA algorithm generated in 35% of cases better solutions compared to the SCA algorithm. Finally, the results show that the difference in the execution time of the algorithms for all graphs is acceptable, but the quality of the solutions generated by the WPA algorithm in some cases is better. Further research is also needed on whether the performance of both algorithms can be improved if other graph data representations are used. For example, if adjacency lists are used to represent graphs, the required memory is 2 m, instead of using an adjacency matrix where the required memory is constant and equal to n^2 (n is the number of vertices in the graph, and m is the number of edges).

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, 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] B. L. Natarajan, "Computation of chromatic numbers for new class of graphs and its applications," *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 8, no. 8, pp. 396–400, 2019.

[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, Sep. 2020, doi: 10.1016/j.disc.2020.111959.

[7] D. S. Malyshev and O. O. Lobanova, "Two complexity results for the vertex coloring problem," *Discrete Applied Mathematics*, vol. 219, pp. 158–166, Mar. 2017, doi: 10.1016/j.dam.2016.10.025.

[8] 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.

[9] Z. Huanping, Z. Peijin, L. Jingwen, and S. Huojie, "A 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.

[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, Dec. 2020, doi: 10.1016/j.entcs.2020.10.007.

[11] 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.

[12] A. M. Foley, D. J. Fraser, C. T. Hoàng, K. Holmes, and T. P. LaMantia, "The intersection of two vertex coloring problems," *Graphs and Combinatorics*, vol. 36, no. 1, pp. 125–138, Jan. 2020, doi: 10.1007/s00373-019-02123-1.

[13] V. Kralev and R. Kraleva, "Methods for software visualization of large graph data structures," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 10, no. 1, Feb. 2020, doi: 10.18517/ijaseit.10.1.10739.

[14] M. Adegbindin, A. Hertz, and M. Bellaïche, "A new efficient RLF-like algorithm for the vertex coloring problem," *Yugoslav Journal of Operations Research*, vol. 26, no. 4, pp. 441–456, 2016, doi: 10.2298/YJOR151102003A.

[15] C. Contreras Bolton, G. Gatica, and V. Parada, "Automatically generated algorithms for the vertex coloring problem," *PLoS ONE*, vol. 8, no. 3, Mar. 2013, doi: 10.1371/journal.pone.0058551.

[16] 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, Dec. 2020, doi: 10.1002/jgt.22572.

[17] 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.

[18] V. Kralev, "Different applications of the genetic mutation operator for symetric travelling salesman problem," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 8, no. 3, pp. 762–770, Jun. 2018, doi: 10.18517/ijaseit.8.3.4867.

[19] V. Kralev, R. Kraleva, V. Ankov, and D. Chakalov, "An analysis between exact and approximate algorithms for the k-center problem in graphs," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 12, no. 2, pp. 2058–2065, Apr. 2022, doi: 10.11591/ijece.v12i2.pp2058-2065.

[20] 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.

[21] 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, Jul. 2021, doi: 10.15807/jorsj.64.184.

[22] 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, Mar. 2021, doi: 10.1142/S0218216521500152.

[23] J. B. Kelly and L. M. Kelly, "Paths and circuits in critical graphs," *American Journal of Mathematics*, vol. 76, no. 4, pp. 786–792, Oct. 1954, doi: 10.2307/2372652.

[24] D. B. West, *Introduction to graph theory*. Pearson College Div, Subsequent edition, 2000.
[25] J. Mitchem, "On various algorithms for estimating the chromatic number of a graph," *The Computer Journal*, vol. 19, no. 2, pp. 182–183, May 1976, doi: 10.1093/comjnl/19.2.182.
[26] O. Borodin and A. Kostochka, "On an upper bound of a graph's chromatic number, depending on the graph's degree and density," *Journal of Combinatorial Theory, Series B*, vol. 23, no. 2–3, pp. 247–250, Oct. 1977, doi: 10.1016/0095-8956(77)90037-5.
[27] A. V. Kostochka, L. Rabern, and M. Stiebitz, "Graphs with chromatic number close to maximum degree," *Discrete Mathematics*, vol. 312, no. 6, pp. 1273–1281, Mar. 2012, doi: 10.1016/j.disc.2011.12.014.
[28] D. J. A. Welsh, "An upper bound for the chromatic number of a graph and its application to timetabling problems," *The Computer Journal*, vol. 10, no. 1, pp. 85–86, Jan. 1967, doi: 10.1093/comjnl/10.1.85.
[29] 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.
[30] B. Boz and G. Sungu, "Integrated crossover based evolutionary algorithm for coloring vertex-weighted graphs," *IEEE Access*, vol. 8, pp. 126743–126759, 2020, doi: 10.1109/ACCESS.2020.3008886.

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 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 rady_kraleva@swu.bg.