

# An analysis between the Welsh-Powell and DSatur algorithms for coloring of sparse graphs

Radoslava Kraleva, Velin Kralev, Toma Katsarski

Department of Informatics, South-West University "Neofit Rilski", Blagoevgrad, Bulgaria

## Article Info

### Article history:

Received Aug 22, 2024

Revised Mar 20, 2025

Accepted May 23, 2025

### Keywords:

Chromatic number

Graph coloring

Graph theory

Heuristic algorithm

Vertex coloring

## ABSTRACT

In this research an analysis between the Welsh-Powell and DSatur algorithms for the graph vertex coloring problem was presented. Both algorithms were implemented and analyzed as well. The method of the experiment was discussed and the 46 test graphs, which were divided into two sets, were presented. The results show that for sparse graphs with a smaller number of vertices and edges, both algorithms can be used for solving the problem. The results show that in 50% of the cases the Welsh-Powell algorithm found better solutions (23 in total). So, the DSatur algorithm found better solutions in only 19.6% of cases (9 in total). In the remaining 30.4% of cases, both algorithms found identical solutions. For graphs with a larger number of vertices, the usage of the Welsh-Powell algorithm is recommended as it finds better solutions. The execution time of the DSatur algorithm is greater than the execution time of the Welsh-Powell algorithm, reaching up to a minute for graphs with a larger number of vertices. For graphs with fewer vertices and edges, the execution times of both algorithms are shorter, but the time is still greater for the DSatur algorithm.

*This is an open access article under the [CC BY-SA](#) license.*



## Corresponding Author:

Radoslava Kraleva

Department of Informatics, Faculty of Mathematics and Natural Science, South-West University

66 Ivan Michailov str., 2700 Blagoevgrad, Bulgaria

Email: rady\_kraleva@swu.bg

## 1. INTRODUCTION

The graph-type structure and the algorithms applied to this structure have been widely and thoroughly researched in the areas of discrete mathematics and computer science in the last few decades [1]. These structures are used in the modeling of many processes in many different fields of science and practice [2]. This makes them useful and often used as a means of analyzing activities, events, and interactions between different objects in different domains of the real world [3]. The graph-type structures are used to present different complex problems that occur often. Then they are studied and analyzed by specialized software applications that execute different algorithms on these structures [4]. Therefore, much research in recent years has been aimed at improving and optimizing various algorithmic methods and approaches for solving different classes of problems modeled by using graph structures [5].

A graph-type structure is a set that has two subsets – a set of vertices (with  $n$  elements) and a set of edges (with  $m$  elements). A vertex set must contain at least one element for the graph itself to make sense. Unlike the set of vertices, the set of edges may not have any elements. This case is not typical of a graph-type structure, although it is possible since edges represent connections between different pairs of vertices. Therefore, the main idea is to use the non-linear data structure, where some objects (vertices) interact (are mutually dependent) in a certain way with each other, and accordingly, these interactions are represented by

edges [6]. If each edge is assigned a certain numerical value, then the graph-type structure is called weighted [7].

The problem of coloring vertices in a graph-type structure is an NP-hard problem [8]. However, due to the extreme applicability and importance of this problem, it continues to be actively researched [9]. Therefore, there are also many different variants (as definitions) of this problem. For example, the vertex coloring with communication constraints in synchronous broadcast networks [10], the reconfiguration graph for vertex colorings of weakly chordal graphs [11], the facial unique-maximum colorings of plane graphs with restriction on big vertices [12], and many others. Separate ways to solve this problem use different algorithms [13], different techniques [14], and different approaches [15]. These methods for solving this problem have also been used in solving other problems modeled by graph-type structures [16]. A more comprehensive presentation of the graph vertex coloring problem is also presented in other scientific works [17]–[20].

Any graph that is not acyclic, not complete, and does not have a cycle of odd length can be colored with several colors that are equal to or less than the largest degree of a vertex in that graph. Proof of this statement is presented in [21]. If  $\chi(G)$  denotes the chromatic number of a given graph, and  $\Delta(G)$  denotes the largest degree of a vertex in this graph, then the above statement can be presented in the following way:  $\chi(G) \leq \Delta(G)$ . Moreover, if for all vertices in a given graph structure, it is true that the number of edges incident to these vertices is greater than 2, i.e. every vertex in this graph structure has degree greater than 2, then the number of colors required to color the given graph will be one greater than the largest degree of a vertex in the same graph, and only if in that graph there is a clique that contains exactly  $\Delta(G) + 1$  vertices [21]. Other results related to the problem of finding the chromatic number of a given graph, as well as finding lower bounds for this number, are presented in [22]. There are many other algorithms (and approaches) for finding the chromatic numbers of graphs, which are presented in detail in various scientific publications [23]–[25].

In this paper, two approximate (heuristic) algorithms for solving the graph vertex coloring problem will be analyzed, respectively – DSatur [26] and Welsh-Powell [27]. These algorithms are approximate and do not always find an optimal solution in terms of the number of generations of the chromatic classes. This means that the vertices of a given graph divide into independent subsets of vertices. This study aims to decide which of the two algorithms will find better solutions for graphs with a predefined number of vertices and edges.

## 2. RESEARCH METHOD

This section presents implementations of the Welsh-Powell and DSatur algorithms. Both algorithms are approximate and used to solve the graph vertex coloring problem. A part of global parameters and arrays must be pre-declared for both algorithms. They are presented in Figure 1.

```

01 var
02   CountOfVertices: Integer;
03   MinimalColorCount: Integer;
04   VectorOfColors: array of TColor;
05   AdjacencyStructure: array of array of Integer;
06   VertexList: array of Integer; EdgeList: array of Integer;

```

Figure 1. Pre-declared global parameters and arrays

The *CountOfVertices* parameter (declared on line 2) contains the number of vertices in the graph data structure. The *MinimalColorCount* parameter (declared on line 3) is an additional parameter used by the algorithms in the decision process. The *VectorOfColors* vector (of type *TColor*), declared on line 3, is used by algorithms to store the array of assorted colors. Each item in this vector stores a color with which the given vertex of the graph data structure is colored. Each structure of the graph type is represented by an adjacency matrix structure (declared on line 5), a vertex list vector, and an edge list vector (declared on line 6). Each item (*s*, *f*) of the *AdjacencyStructure* indicates whether the vertices with indices *s* (start) and *f* (final) are adjacent or not.

The *WelshPowellAlgorithm* procedure implements the first heuristic algorithm for coloring the vertices of a graph. The source code of this procedure is shown in Figure 2. It uses additional (local) parameters - *ColorCount*, *ColoredVerticesCount*, *AcceptDecision*, *Iteration*, and *Vertex*. The *Iteration* and *Vertex* parameters are used for iteration variables of the two *for* loops executed on lines 9 and 14. The

*ColorCount* parameter stores the number of one of the colors used so far. The *AcceptDecision* parameter (of *Boolean* type) shows whether the given vertex can be colored with one of the available colors or not.

The parameter *ColoredVerticesCount* stores the current number of colored vertices in the graph. In the WelshPowell algorithm, the parameters *MinimalColorCount*, *ColorCount*, *ColoredVerticesCount* are set to 0 (lines 3-5). Accordingly, the *AcceptDecision* variable is set to *False* (line 6).

```

01 procedure WelshPowellAlgorithm;
02 begin
03   MinimalColorCount := 0;
04   var ColorCount := 0;
05   var ColoredVerticesCount := 0;
06   var AcceptDecision := False;
07   repeat
08     ColorCount := ColorCount + 1;
09     for var Iteration := 1 to CountOfVertices do
10       begin
11         if (VectorOfColors[Iteration] = 0) then
12           begin
13             AcceptDecision := True;
14             for var Vertex := 1 to CountOfVertices do
15               begin
16                 if ((AdjacencyStructure[Iteration][Vertex] > 0) and
17                     (VectorOfColors[Vertex] = ColorCount)) then
18                   begin
19                     AcceptDecision := False;
20                     Break;
21                   end;
22                 end;
23             if (AcceptDecision = True) then
24               begin
25                 VectorOfColors[Iteration] := ColorCount;
26                 if (MinimalColorCount < ColorCount) then
27                   MinimalColorCount := ColorCount;
28                 ColoredVerticesCount := ColoredVerticesCount + 1;
29               end;
30             end;
31           end;
32   until (ColoredVerticesCount = CountOfVertices);
33 end;

```

Figure 2. Source code of the WelshPowellAlgorithm procedure

The algorithm is executed until all vertices in the graph are colored (line 32 - the “until” condition for the end of the repeat loop). At the beginning of each iteration of the repeat loop, a new color index (number) is selected (line 8). At this step of the algorithm's execution, a traversal of the vertices of the graph begins (line 9). If the currently analyzed vertex is not colored (see line 11) and that vertex has no adjacent vertices that are colored with that color, then the current vertex is colored with the current color (line 25). If among the adjacency vertices of the current vertex is not colored with the current color, then the logical variable *AcceptDecision* will have the value *True*. Only in this case will the current vertex be colored with the current color. The source code of line 26 checks whether the parameter *ColorCount* is greater than the value of the parameter *MinimalColorCount*. If this is the case, then the value of the *ColorCount* parameter is assigned as the value of the *MinimalColorCount* parameter. As another vertex has been successfully colored, this is recorded by updating the value of the *ColoredVerticesCount* parameter, which is incremented by 1. The computational complexity of this algorithm is quadratic and depends on the number of vertices of the graph (the *CountOfVertices* parameter). A similar implementation of this algorithm is presented in [20].

The *DSaturAlgorithm* procedure implements the second heuristic algorithm for coloring the vertices of a graph. The source code of this procedure is shown in Figure 3. It also uses additional parameters – *ColorCount*, *ColoredVerticesCount*, *AcceptDecision*, and *Vertex*. The *Vertex* parameter stores the number (index) of the next vertex that will be colored. The *ColorCount* parameter stores the number of one of the colors used so far.

The parameter *ColoredVerticesCount* stores the current number of colored vertices in the graph. In the DSatur algorithm, the parameters *MinimalColorCount*, *Vertex*, *ColorCount*, and *ColoredVerticesCount* are set to 0 (lines 3-6). The algorithm is executed until all vertices in the graph are colored (line 7 - the while

condition for the beginning of the while loop). At the beginning of each iteration of the while loop, the next index (number) of the vertex is selected (line 9).

The *FoundNextVertex* function selects the next vertex to be colored. This vertex must fulfill one of the following four criteria (in the order in which they are listed). It must have the highest value of the degree of saturation characteristic. In the case that there is more than one vertex with such a value, a second selection criterion is used. According to it, the vertex must have the largest residual degree in the graph induced by the current one after removing the colored vertices. If there are more vertices, a third selection criterion is used. According to it, that vertex is selected, which can be colored with a color with a smaller number. In the last case, the vertex with the smallest number is selected among all vertices having the same values of the above three criteria. At this step, the chosen vertex (indexed by the *Vertex* parameter) is colored with the current color. The value of the *ColoredVerticesCount* parameter is incremented by one after the successful coloring of the vertex (line 13).

```

01 procedure DSaturAlgorithm;
02 begin
03   MinimalColorCount := 0;
04   var Vertex := 0;
05   var ColorCount := 0;
06   var ColoredVerticesCount := 0;
07   while not (ColoredVerticesCount = CountOfVertices) do
08   begin
09     Vertex := FoundNextVertex;
10     if (Vertex > 0) then
11     begin
12       VectorOfColors[Vertex] := ColorCount;
13       ColoredVerticesCount := ColoredVerticesCount + 1;
14       UpdateParameters();
15       UpdateColorCount(ColorCount);
16       if (MinimalColorCount < ColorCount) then
17         MinimalColorCount := ColorCount;
18     end;
19   end;
20 end;

```

Figure 3. Source code of the DSaturAlgorithm procedure

The *UpdateParameters* procedure updates the values of the degree of saturation and residual degree parameters (line 14). The *UpdateColorCount* procedure, which receives an input parameter – *ColorCount* (passed by address), increments the number of available colors if necessary (line 14). The source code of line 16 checks whether the parameter *ColorCount* is greater than the value of the parameter *MinimalColorCount*. In this case, the value of the *ColorCount* parameter is assigned as the value of the *MinimalColorCount* parameter. The computational complexity of this algorithm is also quadratic and depends on the number of vertices of the graph (*CountOfVertices* parameter).

### 3. RESULTS AND DISCUSSION

The results of the experiment are shown and discussed. For this study, 46 sparse graphs, respectively with 800, 1000, 1200, ..., 9600, and 9800 vertices were created. Each graph has a density of 3%. Based on this characteristic of the graph exactly 3% of among all vertices, are randomly selected. These graphs are presented in Table 1. More information, which includes the results of the execution of the two heuristic algorithms, in terms of their execution time and the quality of the generated solutions is shown in Table 2. The experimental conditions are: 64-bit Windows 11 and hardware configuration: Processor: Intel (R) Core (TM) i5-12450H at 4.40 GHz; RAM: 16 GB, SSD 1000 GB.

In Tables 1 and 2, the "*Graph ID*" column shows the number of the test graph. The "*Graph file name*" column shows the name of the file in which the information for the corresponding test graph is stored. The "*Vertex count*" column shows the number of vertices of a graph. The "*Max edge count*" column shows the 100% density of a graph in terms of the number of edges that it would have in the case of the completed graph. The "*Edge count*" column shows the number of edges of the graph. The "*Color*" columns show the required number of colors used in the coloring process. Accordingly, the "*Time (ms)*" columns show the execution time of each of the algorithms.

Table 1. The set of graphs

Graph ID	Graph file name	Vertex count	Max edge count	Edge count	Graph ID	Graph file name	Vertex count	Max edge count	Edge count
1	G_800_9588.gff	800	319 600	9 588	24	G_5400_437320.gff	5 400	14 577 300	437 320
2	G_1000_14986.gff	1 000	499 500	14 986	25	G_5600_470316.gff	5 600	15 677 200	470 316
3	G_1200_21582.gff	1 200	719 400	21 582	26	G_5800_504514.gff	5 800	16 817 100	504 514
4	G_1400_29380.gff	1 400	979 300	29 380	27	G_6000_539910.gff	6 000	17 997 000	539 910
5	G_1600_38376.gff	1 600	1 279 200	38 376	28	G_6200_576508.gff	6 200	19 216 900	576 508
6	G_1800_48574.gff	1 800	1 619 100	48 574	29	G_6400_614304.gff	6 400	20 476 800	614 304
7	G_2000_59970.gff	2 000	1 999 000	59 970	30	G_6600_653302.gff	6 600	21 776 700	653 302
8	G_2200_72568.gff	2 200	2 418 900	72 568	31	G_6800_693498.gff	6 800	23 116 600	693 498
9	G_2400_86364.gff	2 400	2 878 800	86 364	32	G_7000_734896.gff	7 000	24 496 500	734 896
10	G_2600_101362.gff	2 600	3 378 700	101 362	33	G_7200_777492.gff	7 200	25 916 400	777 492
11	G_2800_117558.gff	2 800	3 918 600	117 558	34	G_7400_821290.gff	7 400	27 376 300	821 290
12	G_3000_134956.gff	3 000	4 498 500	134 956	35	G_7600_866286.gff	7 600	28 876 200	866 286
13	G_3200_153552.gff	3 200	5 118 400	153 552	36	G_7800_912484.gff	7 800	30 416 100	912 484
14	G_3400_173350.gff	3 400	5 778 300	173 350	37	G_8000_959880.gff	8 000	31 996 000	959 880
15	G_3600_194346.gff	3 600	6 478 200	194 346	38	G_8200_1008478.gff	8 200	33 615 900	1 008 478
16	G_3800_216544.gff	3 800	7 218 100	216 544	39	G_8400_1058274.gff	8 400	35 275 800	1 058 274
17	G_4000_239940.gff	4 000	7 998 000	239 940	40	G_8600_1109272.gff	8 600	36 975 700	1 109 272
18	G_4200_264538.gff	4 200	8 817 900	264 538	41	G_8800_1161468.gff	8 800	38 715 600	1 161 468
19	G_4400_290334.gff	4 400	9 677 800	290 334	42	G_9000_1214866.gff	9 000	40 495 500	1 214 866
20	G_4600_317332.gff	4 600	10 577 700	317 332	43	G_9200_1269462.gff	9 200	42 315 400	1 269 462
21	G_4800_345528.gff	4 800	11 517 600	345 528	44	G_9400_1325260.gff	9 400	44 175 300	1 325 260
22	G_5000_374926.gff	5 000	12 497 500	374 926	45	G_9600_1382256.gff	9 600	46 075 200	1 382 256
23	G_5200_405522.gff	5 200	13 517 400	405 522	46	G_9800_1440454.gff	9 800	48 015 100	1 440 454

Table 2. Results of the heuristic algorithms for all graphs

Graph ID	Vertex count	Edge count	Welsh-Powell		DSatur		Graph ID	Vertex count	Edge count	Welsh-Powell		DSatur	
			Colors	Time (ms)	Colors	Time (ms)				Colors	Time (ms)	Colors	Time (ms)
1	800	9 588	10	31	9	250	24	5 400	437 320	32	1360	32	16 172
2	1 000	14 986	12	32	11	375	25	5 600	470 316	32	1531	34	17 781
3	1 200	21 582	13	47	12	563	26	5 800	504 514	33	1672	34	19 390
4	1 400	29 380	14	79	12	765	27	6 000	539 910	35	1828	35	21 156
5	1 600	38 376	15	93	13	1 078	28	6 200	576 508	35	1969	35	23 250
6	1 800	48 574	16	125	14	1 375	29	6 400	614 304	36	2187	36	25 515
7	2 000	59 970	16	141	16	1 672	30	6 600	653 302	37	2344	37	27 312
8	2 200	72 568	18	172	17	2 078	31	6 800	693 498	38	2562	38	29 250
9	2 400	86 364	19	203	18	2 641	32	7 000	734 896	39	2719	38	31 875
10	2 600	101 362	20	266	18	2 937	33	7 200	777 492	40	2859	39	35 219
11	2 800	117 558	21	297	20	3 531	34	7 400	821 290	40	3140	39	36 953
12	3 000	134 956	21	360	21	4 000	35	7 600	866 286	40	3312	41	40 828
13	3 200	153 552	23	406	22	4 797	36	7 800	912 484	41	3578	42	42 547
14	3 400	173 350	23	468	22	5 406	37	8 000	959 880	42	3765	43	45 844
15	3 600	194 346	25	516	24	6 187	38	8 200	1 008 478	43	3937	44	48 312
16	3 800	216 544	26	594	24	7 156	39	8 400	1 058 274	43	4359	44	51 250
17	4 000	239 940	26	640	25	8 125	40	8 600	1 109 272	45	4609	45	54 313
18	4 200	264 538	27	750	27	9 032	41	8 800	1 161 468	44	4985	45	58 969
19	4 400	290 334	28	813	27	10 000	42	9 000	1 214 866	47	5188	46	62 328
20	4 600	317 332	29	922	28	11 031	43	9 200	1 269 462	47	5516	47	66 157
21	4 800	345 528	30	1000	30	12 265	44	9 400	1 325 260	48	6094	47	69 938
22	5 000	374 926	30	1172	30	13 578	45	9 600	1 382 256	48	6125	48	76 610
23	5 200	405 522	32	1281	31	15 063	46	9 800	1 440 454	49	6547	50	79 172

Table 2 and the charts in Figures 4, 5, and 6 show the results of the two heuristic algorithms for the graphs presented in Table 1. The results in the columns "Color" and "Time (ms)" (for both Welsh-Powell and DSatur algorithms) show the number of colors needed (more precisely, the number of chromatic classes) to color the corresponding graph, as well as the time to find an acceptable solution. The results show that in the first half of the set of graphs (1 – 23) in only five out of a total of 23 cases (for graphs 7, 12, 18, 21, and 22) the Welsh-Powell algorithm found the same solutions as the DSatur algorithm.

Table 2 and the charts in Figures 4 and 5 show that for the complete set of graphs in only 14 cases, both algorithms found identical solutions. However, the Welsh-Powell algorithm found better solutions in 23 cases. In comparison, the DSatur algorithm found better solutions in only nine cases, but for graphs with a larger number of vertices and edges (in the second half of the graph set).

For all other cases (for this set of graphs), both algorithms found identical solutions. In the second half of the set of graphs (24 – 46) this trend changes. For these graphs, in only five of the cases (graphs 32, 33, 34, 42, and 44) the DSatur algorithm found better solutions than the Welsh-Powell algorithm.

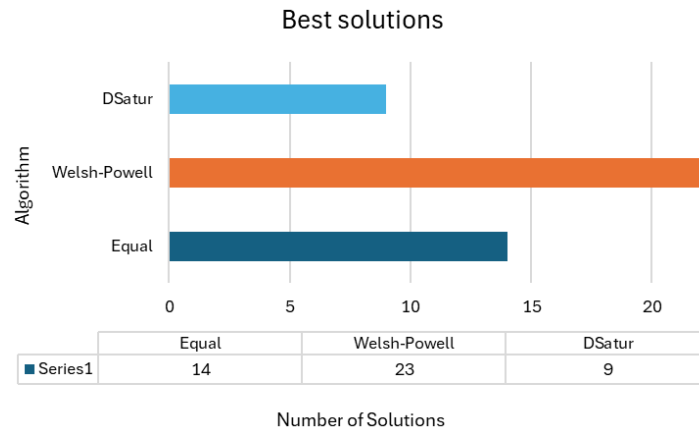


Figure 4. Best solutions found by both algorithms

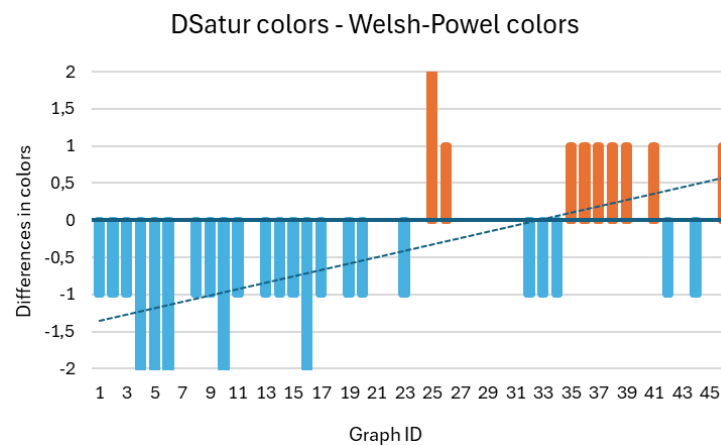


Figure 5. Difference in colors between the DSatur algorithm and the Welsh-Powell algorithm

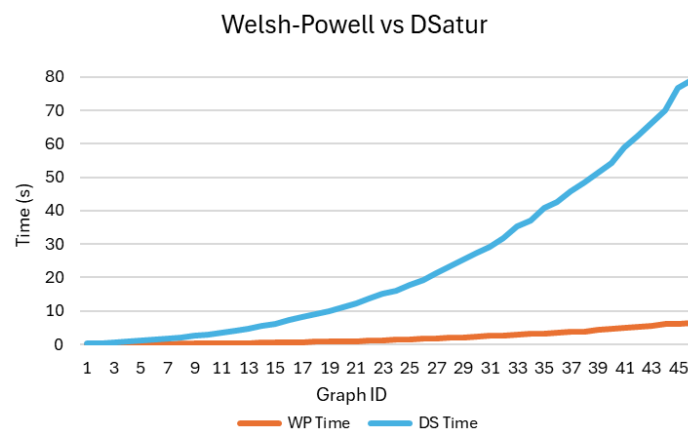


Figure 6. Comparison of the execution times of both algorithms

In contrast, the Welsh-Powell algorithm found better solutions in nine other cases (for graphs 25, 26, 35, 36, 37, 38, 39, 41, and 46). In the remaining nine cases from the second half of the set of graphs, both algorithms found identical solutions (for graphs 24, 27, 28, 29, 30, 31, 40, 43, and 45). From the chart in Figure 5, when the number of vertices in the graph increases, respectively, when the number of edges in the

graph increases, so do the times when the Welsh-Powell algorithm finds better solutions. This trend can be seen from the chart in Figure 5, which shows the difference in solutions found (as the needed number of colors) between both DSatur and Welsh-Powell algorithms. Positive values show better solutions found by the Welsh-Powell algorithm. So, negative values show better solutions found by the DSatur algorithm.

The chart in Figure 6 shows the effect of increasing the size of the graphs (i.e., increasing the number of vertices and the number of edges) on the execution time of both algorithms. The execution time of the DSatur algorithm is greater than the execution time of the Welsh-Powell algorithm, reaching up to a minute for graphs with a larger number of vertices. For example, for graph 46, the execution time of the DSatur algorithm is 79 seconds, and the execution time of the Welsh-Powell algorithm is only 6.5 seconds. The difference between these values is approximately 12 times. For graphs with fewer vertices and edges (from the first half of the graph set), the execution times of both algorithms are smaller, but the difference is again many times larger for the DSatur algorithm.

#### 4. CONCLUSION

In this research, an analysis between two algorithms Welsh-Powell and DSatur, used for coloring sparse graphs, was presented. Some earlier work in the field of graph theory, related to solving this problem, was also discussed. Two approximate algorithms - the Welsh-Powell algorithm and the DSatur algorithm, were implemented and analyzed. The global declarations of different parameters used by both algorithms (variables and arrays) were shown. The source code of the approximate algorithms was presented and analyzed. Due to the multitasking work of the operating system, the run time of both algorithms was recalculated as the average of five starts for each of the 46 analyzed graphs.

The results show that in the first half of the set of graphs (1–23) in only five out of a total of 23 cases, the Welsh-Powell algorithm found the same solutions as the DSatur algorithm. For all other cases, both algorithms found identical solutions. This trend changed for the second half of the graphs set (24–46). In only five cases of the graphs studied, the DSatur algorithm found better solutions than the Welsh-Powell algorithm. In contrast, the Welsh-Powell algorithm found better solutions in nine other cases. Both algorithms found identical solutions in the remaining nine cases from the second half of the graphs set.

In summary, both algorithms found identical solutions for the complete set of graphs in only 14 cases. However, the Welsh-Powell algorithm found better solutions in 23 other cases. In comparison, the DSatur algorithm found better solutions in only nine cases, but for graphs with a larger number of vertices and edges. When the number of vertices (respectively, the number of edges) in the graph increases, the Welsh-Powell algorithm starts to find better solutions more often than the DSatur algorithm. About the quality of the solutions generated (by both algorithms), the Welsh-Powell algorithm found better solutions in 50% of the cases (23 in total). So, the DSatur algorithm found better solutions in only 19.6% of cases (9 in total). In the remaining 30.4% of cases, both algorithms found identical solutions. The results show the effect of increasing the size of the graphs on the execution time of both algorithms. The execution time of the DSatur algorithm is greater than the execution time of the Welsh-Powell algorithm, reaching up to a minute for graphs with a larger number of vertices. For graphs with fewer vertices and edges (from the first half of the graph set), the execution times of both algorithms are shorter, but the difference is again many times larger for the DSatur algorithm. The study can be continued by performing a comparative analysis of the performance of both algorithms if the analyzed graphs are represented by other data structures, such as adjacency lists.

#### 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
Radoslava Krалева	✓	✓		✓	✓	✓	✓		✓	✓		✓	✓	✓
Velin Krалев	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓		✓	✓
Toma Katsarski								✓						



C : Conceptualization	I : Investigation	Vi : Visualization
M : Methodology	R : Resources	Su : Supervision
So : Software	D : Data Curation	P : Project administration
Va : Validation	O : Writing - Original Draft	Fu : Funding acquisition
Fo : Formal analysis	E : Writing - Review & Editing	

## 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, RK, upon reasonable request.

## REFERENCES




- [1] J. Huang, X. Geng, S. Li, and Z. Zhou, "On spectral extrema of graphs with given order and dissociation number," *Discrete Applied Mathematics*, vol. 342, pp. 368–380, Jan. 2024, doi: 10.1016/j.dam.2023.10.001.
- [2] X. Zhang, S. Zhang, C. Ye, and B. Yao, "Graphic lattices made by graph felicitous-type labelings and colorings of topological coding," *Discrete Applied Mathematics*, vol. 336, pp. 37–46, Sep. 2023, doi: 10.1016/j.dam.2023.03.023.
- [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. Slamini, 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.
- [5] Z. Cao, Q. Zhang, J. Guang, S. Wu, Z. Hu, and J. Liu, "SemanticTopoLoop: Semantic loop closure with 3D topological graph based on quadric-level object map," *IEEE Robotics and Automation Letters*, vol. 9, no. 5, pp. 4257–4264, May 2024, doi: 10.1109/LRA.2024.3374169.
- [6] M. Gan and C. Tan, "Mining multiple sequential patterns through multi-graph representation for next point-of-interest recommendation," *World Wide Web*, vol. 26, no. 4, pp. 1345–1370, Jul. 2023, doi: 10.1007/s11280-022-01094-3.
- [7] K. D. Rangaswamy and M. Gurusamy, "Application of graph theory concepts in computer networks and its suitability for the resource provisioning issues in cloud computing-A review," *Journal of Computer Science*, vol. 14, no. 2, pp. 163–172, Feb. 2018, doi: 10.3844/jcssp.2018.163.172.
- [8] M. Theunis and M. Roeloffzen, "Experimental analysis of algorithms for the dynamic graph coloring problem," *Journal of Graph Algorithms and Applications*, vol. 28, no. 1, pp. 313–349, Aug. 2024, doi: 10.7155/jgaa.v28i1.2956.
- [9] C. Konrad and V. Zamaraev, "Distributed minimum vertex coloring and maximum independent set in chordal graphs," *Theoretical Computer Science*, vol. 922, pp. 486–502, Jun. 2022, doi: 10.1016/j.tcs.2022.04.047.
- [10] H. Lakhlef, M. Raynal, and F. Taiani, "Vertex coloring with communication constraints in synchronous broadcast networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 7, pp. 1672–1686, Jul. 2019, doi: 10.1109/TPDS.2018.2889688.
- [11] C. Feghali and J. Fiala, "Reconfiguration graph for vertex colourings of weakly chordal graphs," *Discrete Mathematics*, vol. 343, no. 3, p. 111733, Mar. 2020, doi: 10.1016/j.disc.2019.111733.
- [12] B. Lidický, K. Messerschmidt, and R. Škrekovski, "Facial unique-maximum colorings of plane graphs with restriction on big vertices," *Discrete Mathematics*, vol. 342, no. 9, pp. 2612–2617, Sep. 2019, doi: 10.1016/j.disc.2019.05.029.
- [13] 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.
- [14] K. Adaricheva et al., "Hamilton paths in dominating graphs of trees and cycles," *Graphs and Combinatorics*, vol. 38, no. 6, p. 174, Dec. 2022, doi: 10.1007/s00373-022-02579-8.
- [15] 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.
- [16] 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.
- [17] C. Yang, B. Yao, and Z. Yin, "A new vertex distinguishing total coloring of trees," *AIMS Mathematics*, vol. 6, no. 9, pp. 9468–9475, 2021, doi: 10.3934/math.2021550.
- [18] K. S. Lyngsie and L. Zhong, "Vertex colouring edge weightings: a logarithmic upper bound on weight-choosability," *The Electronic Journal of Combinatorics*, vol. 28, no. 2, Apr. 2021, doi: 10.37236/6878.
- [19] V. Kraleev and R. Kraleva, "An analysis between different algorithms for the graph vertex coloring problem," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 13, no. 3, pp. 2972–2980, Jun. 2023, doi: 10.11591/ijece.v13i3.pp2972-2980.
- [20] 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.
- [21] S. Wang, R. Han, and X. Wang, "A coloring-based packet loss rate measurement scheme on network nodes," *Electronics*, vol. 13, no. 23, p. 4692, Nov. 2024, doi: 10.3390/electronics13234692.
- [22] S. Cambie, "Bounds and monotonicity of critical set parameters of colourings," *Discrete Mathematics*, vol. 347, no. 7, p. 114041, Jul. 2024, doi: 10.1016/j.disc.2024.114041.
- [23] Z. Huanping, X. Dangqin, and S. Huojie, "Strong vertex-distinguishing total coloring algorithm for complete graphs based on equitable coloring," *Journal of Engineering Science and Technology Review*, vol. 13, no. 1, pp. 126–132, 2020, doi: 10.25103/jestr.131.17.
- [24] J. He, Y. Zhang, and S. Lin, "A quantum algorithm for deciding graph 2-coloring problem in embedded systems," *Computer-*






- Aided Design and Applications*, pp. 31–43, Sep. 2023, doi: 10.14733/cadaps.2024.S8.31-43.
- [25] A. Scott, P. Seymour, and S. Spirkl, “Polynomial bounds for chromatic number. IV: a near-polynomial bound for excluding the five-vertex path,” *Combinatorica*, vol. 43, no. 5, pp. 845–852, 2023, doi: 10.1007/s00493-023-00015-w.
- [26] N. Dupin, “Matheuristic variants of DSATUR for the vertex coloring problem,” *Lecture Notes in Computer Science*, vol. 14754, Springer, Cham, pp. 96–111, doi: 10.1007/978-3-031-62922-8\_7 2024.
- [27] D. D. Zaini, H. Vincensius, K. A. N. U. Widjaja, Nurhasanah, and A. T. Handoyo, “Implementing Welsh-Powell algorithm on coloring the map of West Java,” in *Proceedings of the 8th International Conference on Sustainable Information Engineering and Technology*, Oct. 2023, pp. 679–684, doi: 10.1145/3626641.3627210.

## BIOGRAPHIES OF AUTHORS






**Radoslava Kraveva**    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 graphics. She is an editorial board member and reviewer of journals. She can be contacted at email: rady\_kraveva@swu.bg.



**Velin Kravev**    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\_kravev@swu.bg.



**Toma Katsarski**    is an Assistant Professor of Computer Science at the Faculty of Mathematics and Natural Sciences, South-West University “Neofit Rilski”, Blagoevgrad, Bulgaria. He received his M.Sc. degree in computer science from the South-West University, Blagoevgrad, Bulgaria in 2015. He is a Ph.D. student of computer science at the South-West University, Bulgaria. His research interests include optimization problems of graph theory and application development software. He can be contacted at email: t.katsarski@swu.bg.