International Journal of
Intelligent Systems and
Applications in Engineering
IJISAE
ISSN:2147-6799 www.atscience.org/IJISAE
Advanced Technology and Science
Original Research Paper

A Performance Comparison of Graph Coloring Algorithms

Murat Aslan*[1], Nurdan Akhan Baykan[1]

Accepted 3rd September 2016

Abstract: Graph coloring problem (GCP) is getting more popular to solve the problem of coloring the adjacent regions in a map with minimum different number of colors. It is used to solve a variety of real-world problems like map coloring, timetabling and scheduling. Graph coloring is associated with two types of coloring as vertex and edge coloring. The goal of the both types of coloring is to color the whole graph without conflicts. Therefore, adjacent vertices or adjacent edges must be colored with different colors. The number of the least possible colors to be used for GCP is called chromatic number. As the number of vertices or edges in a graph increases, the complexity of the problem also increases. Because of this, each algorithm can not find the chromatic number of the problems and may also be different in their executing times. Due to these constructions, GCP is known an NP-hard problem. Various heuristic and metaheuristic methods have been developed in order to solve the GCP. In this study, we described First Fit (FF), Largest Degree Ordering (LDO), Welsh and Powell (WP), Incidence Degree Ordering (IDO), Degree of Saturation (DSATUR) and Recursive Largest First (RLF) algorithms which have been proposed in the literature for the vertex coloring problem and these algorithms were tested on benchmark graphs provided by DIMACS. The performances of the algorithms were compared as their solution qualities and executing times. Experimental results show that while RLF and DSATUR algorithms are sufficient for the GCP, FF algorithm is generally deficient. WP algorithm finds out the best solution in the shortest time on Register Allocation, CAR, Mycielski, Stanford Miles, Book and Game graphs. On the other hand, RLF algorithm is quite better than the other algorithms on Leighton, Flat, Random (DSJC) and Stanford Queen graphs.

Keywords: *Chromatic number, Graph coloring algorithms.*

1. Introduction

Graph theory is a problem represented with vertices (nodes) and edges (arcs) [1]. Otherwise, graph coloring problem (GCP) is a problem where adjacent vertices or edges in graph must be colored by using different colors [2]. GCP was proposed by Francis Gutrie as the four color problem. Four color problem has described by F. Gutrie to solve the problem of coloring the adjacent regions in a map using the minimum number of diffrent colors [3].
Graph coloring is associated with two types of coloring as vertex and edge coloring [2]. The goal of the both types of coloring is to color the whole graph without conflicts. Therefore, adjacent vertices or edges must be colored with different colors. If there is at least one link (edge) between two nodes, it is called adjacent vertices. If the Fig. 1 examines, it can be seen that there is no edge between V_2 and V_4 vertices. Therefore, these vertices are not adjacent vertices. However, the other vertices in the graph are adjacent because there is at least one edge between each other. In the context of this study we described vertex coloring problem in graphs. If an undirected $= (V, E)$ graph is examining; V is the set of vertices and E is the set of edges. The graph which is given in Fig. 1, the set of vertices are $V = \{v_1, v_2, v_3, v_4\}$ and the set of edges are $E = \{e_1, e_2, e_3, e_4, e_5\}$. In addition, R = $\{1,2, \dots \dots \dots, k\}$ is the set of colors which are used for coloring the vertices. In this case, if the whole graph colored without conflicts is performed by utilizing the minimum number of different colors, it's called "k-coloring graph" [4]. This minimum number of different colors is known as chromatic number. Chromatic number is indicated by $\chi(G)$ [5,6].

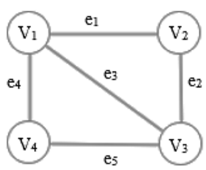

Figure 1. A graph with four vertex and five edge

Graph coloring problem is mostly used for solving computer based applications and problems. Graph coloring algorithms are usable for solving the many engineering applications and real-world problems [2]. Some of the these problems are Map Coloring [7], Timetabling and Scheduling problems [8,9], Register Allocation problems [10,11], Sudoku problem [12] and Frequency Assignment problems [13].
As the number of vertices or edges in a graph increases, the complexity of the problem also increases. Because of this, each algorithm can not find the chromatic number of the problems and may also be different in their executing times. Due to these conditions, GCP is known an NP-hard problem [14]. Hence, for getting a better solution for GCP many hueristic and meta-huestistic algorithms are developed. Huerictic algorithms generally can be used for a problem with fewer numbers of

[1] *Department of Computer Engineering, Engineering Faculty, Selcuk University, Konya, Turkey*
** Corresponding Author: Email: muarataslan@selcuk.edu.tr*
Note: This paper has been presented at the 3rd International Conference on Advanced Technology & Sciences (ICAT'16) held in Konya (Turkey), September 01-03, 2016.

vertices. On the other hand, for the complex graphs meta-hueristic algorithms can find better solutions [15].

Tabu Search (TS) Algorithm [16], Simulated Annealing (SA) Algorithm [17], Genetic Algorithm (GA) [7], Ant Colony (ACO) Algorithm [18], Cuckoo (COA) Algorithm [15] are some of the meta heueristic algorithms used for graph coloring problem. When the vertices in a graph G are colored by means of the greedy algorithms, the coloring issue is performed with selecting and coloring methods of algorithms. These algorithms are called greedy algorithms because of the algorithms choice the best validy selection for every operation step. The greedy algorithms generally provide effective and sufficient results for vertex coloring [15]. In this study, we described First Fit (FF) [19], Largest Degree Ordering (LDO) [19], Welsh and Powell (WP) [5], Incidence Degree Ordering (IDO) [19], Recursive Largest First (RLF) [20] and Degree of Saturation (DSATUR) [21] algorithms which have been proposed in the literature for the vertex coloring problem and these algorithms were tested on benchmark graphs provided by DIMACS [22]. The performances of the algorithms were compared with each other in terms of their solution qualities and executing times.

2. Vertex Coloring Problem

If the vertices in a graph are colored with different colors without considering their adjacencies, this graph would be colored utilizing the number of the different colors which are equal to number of vertices. However, this is not a good solution for GCP. Because, the purpose of the GCP is to find the minimum number of the colors for adjacent vertices colored with different colors.

As the number of vertices or edges in a graph increases, the complexity of the graph also increases. Because of this, coloring the entire graph is getting difficult by the least possible different colors. Therefore, we need some particular methods for coloring the graphs. Thanks to particular methods, the graphs can be colored with minimal different colors.

Algorithms in the literature use the adjacency matrix for coloring the vertices of graphs. Adjacency matrix is generated based on the condition that whether any edge exists between vertices [1]. For a G graph, the set of vertices are shown in the set of $V = \{v_1, v_2, \dots\dots\dots, v_n\}$. The adjacency matrix of a graph is generated by the equation 1. A is represents the adjacency matrix.

$$A = \begin{cases} 1, & \text{if the exists an egde between } V_i \text{ } ile \text{ } V_j \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Another important constraint for the selection of the vertex to be colored is vertex degree. A degree of a vertex in an undirected and unweight graph is equal to the total number of edges connected to the vertex. It's shown with $deg(v_i)$ [1]. The graph given in Fig. 2 has seven vertices and nine edges.

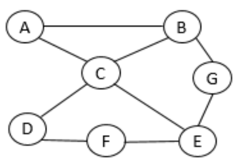

Figure 2. A graph with seven vertices

Table 1 shows the adjacency matrix for the graph presented in Fig. 2 and and Table 2 shows the degrees of vertices for this graph.

Table 1. Adjacency matrix

	A	B	C	D	E	F	G
A	0	1	1	0	0	0	0
B	1	0	1	0	0	0	1
C	1	1	0	1	1	0	0
D	0	0	1	0	0	1	0
E	0	0	1	0	0	1	1
F	0	0	0	1	1	0	0
G	0	1	0	0	1	0	0

Table 2. The degree of vertices

Vertex	A	B	C	D	E	F	G
$Deg(\text{v})$	2	3	4	2	3	2	2

3. Vertex Coloring Algorithms In Graphs

At this section of the study we describe the steps of the some algorithms which have been proposed in the literature for the vertex coloring problem. Also these algorithms will be tested on the graph in Fig. 2. Then the selecting order of the vertices and the color of each vertex are given.

3.1. First Fit Algorithm (FF)

For the given G graph, the set of the vertices is described as $V = \{v_1, v_2, \dots\dots, v_n\}$ and the set of the colors is described as $R = \{r_1, r_2, \dots, r_k\}$. The steps of the algorithm:

Step 1: Create a color set. (initially the color set is empty).
Step 2: The first vertex in the set of V is selected as the starting vertex. The selected vertex is colored with first color and this color is added to color set.
Step 3: Next vertex in the set of V is selected for coloring.
Step 4: For selected vertex, find the adjacent vertices of it from the adjacency matrix. A color which is in the color set, but not color of the adjacent vertices of selected vertex is given to the selected vertex. If the colors in the color set unsuitable for coloring the selected vertex, a new color is defined. The new color is added to the color set and appointed to the selected vertex. If the uncolored vertex exists, it is returned to the step 3.

Table 3 shows the result for the graph shown in Fig. 2. Selected order of the vertices and their colors are given in Table 3. According to Table 3, the first colored vertex is A and the color r_1 is given this vertex. The last colored vertex is G and the color r_3 is given this vertex.

Table 3. The result of the colored graph using the FF algorithm

	A	B	C	D	E	F	G
Selected order	1	2	3	4	5	6	7
Vertex color	r_1	r_2	r_3	r_1	r_1	r_2	r_3

3.2. Welsh Powell Algorithm (WP)

For the given graph $\boldsymbol{G}$, the set of the vertices is described as $\boldsymbol{V} = \{\boldsymbol{v_1}, \boldsymbol{v_2}, \dots\dots, \boldsymbol{v_n}\}$ and the set of the colors of the vertices is described as $\boldsymbol{R} = \{\boldsymbol{r_1}, \boldsymbol{r_2}, \dots, \boldsymbol{r_k}\}$. The steps of the WP algorithm:

Step 1: The vertex degree of each vertex is calculated and the vertex degrees are added to the degree set $Deg(v_i)$, such that $i = 1,2, \dots, n$.
Step 2: The uncolored vertex that has the largest degree in the degree set $Deg(v_i)$ is selected for coloring. Initially, the first color in the color set is selected as the active color.

Step 3: The selected vertex is colored with active color. After that, find the uncolored vertices from adjacency matrix which are not adjacent vertices of the colored vertex and these vertices are added to the V' set ($V' = \{v'_1, v'_2, \ldots, v'_n\}$).

- The uncolored vertex that has the largest degree in the V is selected for coloring. This vertex is colored with active color. After that, the adjacent vertices of the this vertex deleted from V'. This step is repeated until all vertices colored in the set of V'.

Step 4: If the uncolored vertex exists, next color in the color set is selected as active color and it is returned to the step 2. Otherwise the program is terminated, because all vertices in the graph are colored.

Table 4 shows the result for the graph shown in Fig. 2. Selected order of the vertices and their colors are given in Table 4.

Table 4. The result of the colored graph using the WP algorithm

	A	B	C	D	E	F	G
Selected order	7	4	1	6	5	2	3
Vertex color	r_3	r_2	r_1	r_2	r_2	r_1	r_1

3.3. Largest Degree Ordering Algorithm (LDO)

For the given graph $\boldsymbol{G}$, the set of the vertices is described as $\boldsymbol{V} = \{\boldsymbol{v_1}, \boldsymbol{v_2}, \ldots, \boldsymbol{v_n}\}$ and the set of the colors of the vertices is described as $\boldsymbol{R} = \{\boldsymbol{r_1}, \boldsymbol{r_2}, \ldots, \boldsymbol{r_k}\}$. The steps of the LDO algorithm:

Step 1: Create a color set (initially the color set is empty). The vertex degree of each vertex is calculated and the vertex degrees are added to the degree set $Deg(v_i)$, such that $i = 1,2, \ldots, n$.
Step 2: The uncolored vertex that has the largest degree in the degree set $Deg(v_i)$ is selected for coloring. Firstly, the selected vertex is tried to color with the colors in the color set. If the color set is empty or the colors in the color set are not appropriate (all colors int the color set used from the adjacent vertices) for color the vertex, a new color is defined. The new color is added to the color set and appointed to the selected vertex.
Step 3: If the uncolored vertex exists, it is returned to the step 2. Otherwise the program is terminated.
Table 5 shows the result for the graph shown in Fig. 2. Selected order of the vertices and their colors are given in Table 5.

Table 5. The result of the colored graph using the IDO algorithm

	A	B	C	D	E	F	G
Selected order	4	2	1	5	3	6	7
Vertex color	r_3	r_2	r_1	r_2	r_2	r_1	r_1

3.4. Incidence Degree Ordering Algorithm (IDO)

For the given graph G, the set of the vertices is described as $V = \{v_1, v_2, \ldots, v_n\}$ and the set of the colors of the vertices is described as $R = \{r_1, r_2, \ldots, r_k\}$. The steps of the IDO algorithm:

Step 1: The vertex degree of each vertex is calculated and the vertices degrees are added to the degree set $Deg(v_i)$, such that $i = 1,2, \ldots, n$. Initially, there is just one color in the color set.
Step 2: The uncolored vertex that has the largest degree in the degree set $Deg(v_i)$ is selected for coloring. The selected vertex is colored with the first color.
Step 3: The number of the colored adjacent vertices is calculated for every uncolored vertices. After that, the uncolored vertex whose colored neighboring vertices are the maximum is selected. If more than one vertex provides this condition, the vertex which has the largest degree among them is selected.
Step 4: Firstly, the selected vertex with the colors in the color set is tried to color. If the colors in the color set are not appropriate to color the vertex, a new color is defined. The new color is added to the color set and appointed to the selected vertex.
Step 5: If the uncolored vertex exists, it is returned to the step 3. Otherwise, the program is terminated.

Table 6 shows the result for the graph shown in Fig. 2. Selected order of the vertices and their colors are given in Table 6.

Table 6. The result of the colored graph using the IDO algorithm

	A	B	C	D	E	F	G
Selected order	4	3	1	6	2	7	5
Vertex color	r_3	r_2	r_1	r_2	r_2	r_1	r_1

3.5. Degree of Saturation Algorithm (DSATUR)

For the given graph G, the set of the vertices is described as $V = \{v_1, v_2, \ldots, v_n\}$ and the set of the colors of the vertices is described as $R = \{r_1, r_2, \ldots, r_k\}$. The steps of the DSATUR algorithm:

Step 1: The vertex degree of each vertex is calculated and the vertices degrees are added to the degree set $Deg(v_i)$, such that $i = 1,2, \ldots, n$.
Step 2: The uncolored vertex that has the largest degree in the degree set $Deg(v_i)$ is selected for coloring. The selected vertex is colored with first color.
Step 3: Firstly, calculate the number adjacent vertices which are colored with different colors for every uncolored vertex. After that, the uncolored vertex whose number of adjacent vertices colored with different colors is the maximum is selected for coloring. If more than one vertex provide this condition, the vertex which has the largest degree among them is selected.
Step 4: Firstly, the selected vertex is tried to color with the colors in the color set. If the colors in the color set are not appropriate to color the vertex, a new color is defined. The new color is added to the color set and appointed to the selected vertex.
Step 5: If the uncolored vertex exists, it is returned to the step 3. Otherwise the program is terminated.

For DSATUR algorithm; Table 7 shows the result for the graph shown in Fig. 2.

Table 7. The result of the colored graph using the DSATUR algorithm

	A	B	C	D	E	F	G
Selected order	4	3	1	5	2	6	7
Vertex color	r_3	r_2	r_1	r_2	r_2	r_1	r_1

3.6. Recursive Largest First Algorithm (RLF)

RLF is used a recursive structure for coloring the vertices in graph. This recursive structure is the most important feature of the RLF algorithm [20]. According to this recursive structure, whole graph is colored with minimum different colors. For the given G graph, the set of the vertices is described as $V = \{v_1, v_2, \ldots, v_n\}$ and the set of the colors is described as $R = \{r_1, r_2, \ldots, r_k\}$. The steps of the RLF algorithm:

Step 1: Vertex degree is calculated for each vertex and the degrees of vertices added to the set of $Deg(v_i)$. Initially, the first color in the color set is selected as the active color. Select the uncolored vertex which has the largest degree from set of $Deg(v_i)$ for coloring.

Step 2: The selected vertex is colored with active color. Adjacent vertices of the selected vertex can not color with active color. But the uncolored vertices which are not adjacent vertices of the colored vertex can be colored with active color. So RLF uses a recursive structure for select the uncolored vertices to color with active color. During this process the below steps should be followed:

- Adjacent vertices of the selected vertex vi are found from adjacency matrix. Adjacent vertices are added to the adjacent set U. ($U = \{u_1, u_2, \dots \dots, u_t\}$)
- The vertices which are not adjacent vertices of the selected vertex vi are found from adjacency matrix. These vertices are added to the set of V' . Calculate the number adjacent vertices which are in the set of U for every vertex in set of V'. After that, the uncolored vertex whose has maximum adjacent vertices (which are in the set of U) in the set of V' is selected for coloring. The selected vertex is colored with active color.
- The colored vertex and the adjacent vertices of the colored vertex are deleted from V' and added to the set of U.
- If the set of V' is not empty, it is returned to the step 2. Otherwise move to step 3.

Step 3: If the uncolored vertex exists, next color in the color set is selected as active color. Otherwise the program is terminated.

Step 4: Calculate the number adjacent vertices for every uncolored vertex. After that, the uncolored vertex whose has maximum adjacent vertices is selected for coloring process. If more than one vertex provide this condition, the vertex which has the largest degree among them is selected. Then, it is returned to the step 2

For RLF algorithm; Table 8 shows the result for the graph shown in Fig. 2. Selected order of the vertices and their colors are given in Table 8.

Table 8. The result of the colored graph using the RLF algorithm

	A	B	C	D	E	F	G
Selected order	4	7	1	5	6	2	3
Vertex color	r_2	r_3	r_1	r_2	r_2	r_1	r_1

4. Experiment Result

FF, LDO, WP, IDO, DSATUR and RLF algorithms were tested on benchmark graphs provided by DIMACS [22]. The reason of preferring the DIMACS graph, it's given a standard for performance comparison of the algorithms. The performances of the algorithms were compared as their solution quality and executing times. The edge density (D) of the benchmark graphs which are used in this study are calculated from equation 2. E represents the number of the edges and V represents the vertices number of the graph [23].

$$D = \frac{2 * E}{\mathrm{V} * (\mathrm{V} - 1)} \qquad (2)$$

In this study we used Mycielski, CAR, Stanford Graph Base (SGB), Register Allocation, Leighton, Flat, Random (DSJC) and Random geometric (DSJR and R250) DIMACS graphs. V represents the number of the vertices, E is the number of the edges, Den. represents density, Best/χ(G) means chromatic number or the best known number, R represents the number of colors that algorithms are found, T is computation time in seconds. The algorithms are written in the programming language Matlab R2010a. For experiments we used a Laptop computer. It has Intel Core i5 2.20 GHz processor and 8 GB DDR3 RAM.

Mycielski graphs are triangle free graphs. It's mean that the edge connections in the graph must be free of triangle. For mycielski graph, if the vertices in the graph increases, the number of the colors for coloring the graph increases too [24]. Stanford GraphBase (SGB) graphs are created from Donald Knuth. SGB graph can be divided to books, miles, game and queen graphs [22]. For books graphs, a character in the book represents a vertex. So the books graphs are created for holds to relationship between characters. If the characters in the book have relationship to each other, an edge is generated between two vertex which characters run across in the book. These books are Charles Dicken's David Copperfield (david), Victor Hugo's Les Misêrables (jean), Lev Tolstoy's Anna Karenina (anna), Homer's Iliad (homer) and Mark Twain's Huckleberry Finn (huck). For miles graphs the vertices represents some of the United States cities and if there is a road between two cities which provides the conditions, there is an edge generated between them. For game graph, any vertex in the graph represents a college team. There is an edge generated between for every two teams when the teams played to each other during the season. Table 9 shows the results for algorithms which are used for this study. Except the FF algorithm all other algorithms find out the best/χ(G) results. In addition, FF algorithm finds out the best/χ(G) results for graphs which are used for this study except miles graphs and anna, David, homer graph. On the other hand, if the algorithms compared to each other about their computation times, WP algorithm is reached these conclusions in a quite short time.

Table 9. The results and computation times for Mycielski and SGB graphs

Graph	V	E	Den.	Best/χ(G)	RLF		DSATUR		WP		LDO		IDO		FF	
					R	T	R	T	R	T	R	T	R	T	R	T
myciel3	11	20	0,33	4	4	0,0004	4	0,0023	4	**0,0001**	4	0,0002	4	0,0009	4	0,0001
myciel4	23	71	0,27	5	5	0,0009	5	0,0077	5	**0,0002**	5	0,0005	5	0,0028	5	0,0003
myciel5	47	236	0,21	6	6	0,0024	6	0,0255	6	**0,0003**	6	0,0010	7	0,0085	6	0,0006
myciel6	95	755	0,17	7	7	0,0075	7	0,0876	7	**0,0004**	7	0,0024	7	0,0309	7	0,0016
myciel7	191	2360	0,13	8	8	0,0293	8	0,3254	8	**0,0006**	8	0,0068	8	0,1366	8	0,0054
miles1000	128	3216	0,39	42	42	**0,1065**	42	1,2942	43	0,0016	43	0,0123	43	0,7013	44	0,0121
miles1500	128	5198	0,63	73	73	0,3158	73	2,6877	73	**0,0024**	73	0,0219	73	1,6033	76	0,0220
miles500	128	1170	0,14	20	20	0,0250	20	0,3249	20	**0,0010**	20	0,0055	20	0,1360	22	0,0049
miles750	128	2113	0,26	31	31	0,0522	31	0,7112	32	**0,0013**	32	0,0083	31	0,3443	34	0,0081
anna	138	493	0,05	11	11	0,0135	11	0,1231	11	**0,0006**	11	0,0037	11	0,0457	12	0,0026
david	87	406	0,11	11	11	0,0076	11	0,1026	11	**0,0005**	11	0,0025	11	0,0346	12	0,0017

homer	561	1629	0,01	13	13	0,3404	13	0,5412	13	**0,0024**	13	0,0232	13	0,2460	15	0,0183
huck	74	301	0,11	11	11	0,0062	11	0,0745	11	**0,0005**	11	0,0021	11	0,0244	11	0,0014
jean	80	254	0,08	10	10	0,0060	10	0,0616	10	**0,0004**	10	0,0020	10	0,0198	10	0,0013
games120	120	638	0,09	9	9	0,0180	9	0,1537	9	**0,0006**	9	0,0039	9	0,0577	9	0,0032

R: Result of the algorithm, T: Computation time (in second)

Queen graphs are nxn dimensional chessboard graphs. If two queens on the chessboard are in the same row, column, or diagonal, there is an edge generated between them. So, if two queens placed in same row, column or diagonal, one queen can eat the other one. Because of this, there is an edge between them for they don't eat each other. For queens graph if only the graph is colored with minimum number n, two queens can move on chessboard.

Table 10 shows the results for queens graphs. Experimental results show that while RLF and DSATUR algorithms are sufficient for the queens graphs, but the other algorithms are generally deficient. RLF algorithm finds out just only the chromatic number of queen5_5 graph and also it finds out quite better results for the other queen graphs. DSATUR algorithm generally finds out good results, but it is very slow according to the RLF.

Table 10. The results and computation times for Queen graphs

Graph	V	E	Den.	Best/χ(G)	RLF		DSATUR		WP		LDO		IDO		FF	
					R	T	R	T	R	T	R	T	R	T	R	T
queen5_5	25	160	0,51	5	**5**	**0,0012**	**5**	0,0332	7	0,0003	7	0,0006	7	0,0109	8	0,0005
queen6_6	36	290	0,45	7	**8**	**0,0028**	9	0,0634	9	0,0003	9	0,0010	10	0,0218	11	0,0008
queen7_7	49	476	0,40	7	**9**	**0,0045**	11	0,1090	12	0,0005	12	0,0016	12	0,0461	10	0,0013
queen8_12	96	1368	0,30	12	**13**	**0,0202**	14	0,3851	15	0,0007	15	0,0045	15	0,1779	15	0,0041
queen8_8	64	728	0,36	9	**11**	**0,0081**	12	0,1783	13	0,0005	13	0,0024	15	0,0923	13	0,0020
queen9_9	81	1056	0,32	10	**12**	**0,0154**	13	0,2853	15	0,0007	15	0,0036	15	0,1312	16	0,0030
queen10_10	100	2940	0,59	11	**13**	**0,0215**	14	0,4351	17	0,0009	17	0,0052	17	0,1876	16	0,0044
queen11_11	121	3960	0,54	11	**14**	**0,0345**	15	0,6300	17	0,0009	17	0,0072	18	0,2964	17	0,0063
queen12_12	144	5192	0,50	13	**15**	**0,0550**	16	0,9163	19	0,0010	19	0,0100	20	0,4604	20	0,0092
queen13_13	169	6656	0,47	13	**16**	**0,0800**	17	1,3226	23	0,0013	23	0,0134	22	0,6869	21	0,0125
queen14_14	196	8372	0,44	16	**17**	**0,1227**	19	1,8408	25	0,0015	25	0,0170	24	1,0488	23	0,0169

R: Result of the algorithm, T: Computation time (in second)

CAR graphs are created from inspiration of the mycielski graphs. After the some new vertices inserted graph, the graph size increases, but the density of the graph is unchanging [25]. The CAR graphs are more difficult than the mycielski graphs. Table 11 shows the results for CAR graphs. According to the Table 11; RLF, DSATUR, WP and LDO algorithms reach the best/χ(G) results. Furthermore the IDO algorithm generally finds out the best/χ(G) results. But FF algorithm is generally deficient. If the algorithms compare to each other about their computation times, the best algorithm for CAR graphs is WP algorithm. Because WP algorithm reaches the best/χ(G) results shortest computation times.

Table 11. The results and computation times for CAR graphs

Graph	V	E	Den.	Best/χ(G)	RLF		DSATUR		WP		LDO		IDO		FF	
					R	T	R	T	R	T	R	T	R	T	R	T
1_Fullins_4	93	593	0,14	5	5	0,0069	5	0,0869	5	**0,0003**	5	0,0021	6	0,0298	11	0,0016
1_Fullins_5	282	3247	0,08	6	6	0,0612	6	0,5026	6	**0,0007**	6	0,0104	7	0,2167	14	0,0098
1_Insertions_4	67	232	0,10	5	5	0,0058	5	0,0258	5	**0,0002**	5	0,0013	5	0,0090	5	0,0008
1_Insertions_5	202	1227	0,06	6	6	0,0314	6	0,1527	6	**0,0005**	6	0,0055	6	0,0569	6	0,0038
1_Insertions_6	607	6337	0,03	7	7	0,3575	7	1,2288	7	**0,0023**	7	0,0344	7	0,6011	7	0,0308
2_Fullins_3	52	201	0,15	5	5	0,0027	5	0,0237	5	**0,0002**	5	0,0010	5	0,0076	10	0,0008
2_Fullins_4	212	1621	0,07	6	6	0,0329	6	0,2116	6	**0,0006**	6	0,0060	6	0,0796	14	0,0052
2_Fullins_5	852	12201	0,03	7	7	0,8307	7	3,2494	7	**0,0044**	7	0,0703	7	1,8256	18	0,0763
2_Insertions_4	149	541	0,05	5	5	0,0188	5	0,0621	5	**0,0004**	5	0,0036	5	0,0230	5	0,0021
2_Insertions_5	597	3936	0,02	6	6	0,3721	6	0,6322	6	**0,0023**	6	0,0276	6	0,2985	6	0,0211
3_Fullins_3	80	346	0,11	6	6	0,0060	6	0,0380	6	**0,0003**	6	0,0017	6	0,0134	12	0,0012
3_Fullins_4	405	3524	0,04	7	7	0,1476	7	0,5354	7	**0,0012**	7	0,0157	8	0,2370	17	0,0150
3_Fullins_5	2030	33751	0,02	8	8	10,3646	8	18,3988	8	**0,0254**	8	0,3786	9	11,5821	22	0,4600
3_Insertions_3	56	110	0,07	4	4	0,0033	4	0,0125	4	**0,0002**	4	0,0010	4	0,0047	4	0,0006
3_Insertions_4	281	1046	0,03	5	5	0,0731	5	0,1288	5	**0,0007**	5	0,0073	5	0,0498	5	0,0048
3_Insertions_5	1406	9695	0,01	6	6	3,6966	6	2,3609	6	**0,0128**	6	0,1101	7	1,2766	6	0,0996
4_Fullins_3	114	541	0,08	7	7	0,0107	7	0,0610	7	**0,0004**	7	0,0026	7	0,0216	14	0,0020
4_Fullins_4	690	6650	0,03	8	8	0,5299	8	1,2976	8	**0,0031**	8	0,0386	8	0,6677	20	0,0387
4_Fullins_5	4146	77305	0,01	9	9	89,5661	9	85,9533	9	**0,1131**	9	1,6550	9	55,6602	26	2,0289
4_Insertions_3	79	156	0,05	4	4	0,0065	4	0,0180	4	**0,0002**	4	0,0015	4	0,0068	4	0,0009
4_Insertions_4	475	1795	0,02	5	5	0,2527	5	0,2467	5	**0,0015**	5	0,0188	5	0,1009	5	0,0103
5_Fullins_3	154	792	0,07	8	8	0,0220	8	0,0922	8	**0,0005**	8	0,0036	8	0,0332	16	0,0029
5_Fullins_4	1085	11395	0,02	9	9	1,8848	9	2,9627	9	**0,0080**	9	0,0874	9	1,6530	23	0,0923

R: Result of the algorithm, T: Computation time (in second)

Random (DSJ) graphs which are created from David Johnson and R250_5 graph are difficult to solve benchmark graphs [15]. Flat graphs are created from Culberson. [26]. First parameter of the Flat graphs represents the number of the vertices and the second parameter represents the chromatic number.

Table 12 shows the results for Random and Flat graphs. According to the Table 12, RLF and DSATUR algorithms generally find out the best/χ(G) results. For the R250_5 graph, DSATUR algorithm's computation time is further than RLF's. But DSATUR finds out quite better result than RLF for R250_5 graph. On the other hand, RLF finds out quite better results for the other graphs and RLF is faster than the other algorithms.

fpsol2*, inithx*, zeroin* and mulsol* graphs are computer register allocation problem graphs which are generated from Gary Lewandowski [24]. These graphs are real-world problem's graphs. The computer registers and the operations are defined as vertices. If a register and an operation have a relationship, there is an edge generated between them.

Table 13 shows the results for computer register allocation graphs. All algorithms which are used for this study reach the χ(G) results for computer register allocation graphs. If the algorithms compare to each other about their computation times, the best algorithm for register allocation graphs is WP algorithm and also FF algorithm reaches the χ(G) results a quite short times. The slowest algorithm for these graphs is DSATUR algorithm.

Table 12.The results and computation times for Random and Flat graphs

Graf	V	E	Den.	Eniyi/ χ(G)	RLF		DSATUR		WP		LDO		IDO		FF	
					R	T	R	T	R	T	R	T	R	T	R	T
DSJC125_1	125	736	0,09	5	**6**	**0,0135**	**6**	0,0846	7	0,0005	7	0,0032	7	0,0320	8	0,0024
DSJC125_5	125	3891	0,50	17	**21**	**0,0468**	22	0,6111	23	0,0011	23	0,0079	25	0,2966	26	0,0074
DSJC125_9	125	6961	0,89	44	**49**	**0,1811**	51	1,4215	53	0,0019	53	0,0162	54	0,7575	56	0,0154
DSJC250_1	250	3218	0,10	8	**10**	**0,0665**	**10**	0,4791	11	0,0011	11	0,0142	12	0,2086	13	0,0097
DSJC250_5	250	15668	0,50	28	**35**	**0,4661**	37	4,9399	41	0,0025	41	0,0371	40	3,0145	43	0,0394
DSJR500_1	500	3555	0,03	12	**12**	**0,2863**	13	0,5829	13	0,0023	13	0,0237	13	0,2609	15	0,0199
R250_5	250	14849	0,48	65	71	0,7803	**68**	**4,6108**	70	0,0034	70	0,0439	69	2,7790	79	0,0478
flat300_20	300	21375	0,48	20	**38**	**0,7199**	42	8,5125	44	0,0032	44	0,0554	45	5,3253	47	0,0609
flat300_26	300	21633	0,48	26	**39**	**0,8435**	41	8,5990	45	0,0030	45	0,0566	48	5,5501	45	0,0610
flat300_28	300	21695	0,48	28	**38**	**0,8624**	42	8,6611	45	0,0030	45	0,0564	48	5,5324	46	0,0613

R: Result of the algorithm, T: Computation time (in second)

Table 13. The results and computation times for Register Allocation graphs

Graph	V	E	Den.	Best/χ(G)	RLF		DSATUR		WP		LDO		IDO		FF	
					R	T	R	T	R	T	R	T	R	T	R	T
fpsol2_i1	496	11654	0,09	65	65	0,9869	65	3,1791	65	**0,0044**	65	0,0646	65	1,8096	65	0,0552
fpsol2_i2	451	8691	0,09	30	30	0,5217	30	1,9960	30	**0,0024**	30	0,0442	30	1,1139	30	0,0409
fpsol2_i3	425	8688	0,10	30	30	0,5184	30	1,9752	30	**0,0022**	30	0,0427	30	1,0739	30	0,0407
mulsol_i1	197	3925	0,20	49	49	0,1299	49	0,6347	49	**0,0021**	49	0,0153	49	0,2924	49	0,0137
mulsol_i2	188	3885	0,22	31	31	0,1171	31	0,6423	31	**0,0015**	31	0,0145	31	0,2899	31	0,0133
mulsol_i3	184	3916	0,23	31	31	0,1164	31	0,6189	31	**0,0015**	31	0,0143	31	0,2805	31	0,0134
mulsol_i4	185	3946	0,23	31	31	0,1243	31	0,6328	31	**0,0015**	31	0,0145	31	0,2994	31	0,0130
mulsol_i5	186	3973	0,23	31	31	0,1253	31	0,6286	31	**0,0015**	31	0,0145	31	0,2900	31	0,0128
inithx_i1	864	18707	0,05	54	54	2,7427	54	6,7614	54	**0,0066**	54	0,1337	54	4,2802	54	0,1266
inithx_i2	645	13979	0,07	31	31	1,4014	31	4,2319	31	**0,0037**	31	0,0839	31	2,5214	31	0,0800
inithx_i3	621	13969	0,07	31	31	1,3034	31	4,1724	31	**0,0035**	31	0,0819	31	2,5577	31	0,0780
zeroin_i1	211	4100	0,18	49	49	0,1427	49	0,6636	49	**0,0020**	49	0,0157	49	0,3188	49	0,0139
zeroin_i2	211	3541	0,16	30	30	0,1062	30	0,5390	30	**0,0014**	30	0,0136	30	0,2504	30	0,0124
zeroin_i3	206	3540	0,17	30	30	0,1150	30	0,5439	30	**0,0014**	30	0,0134	30	0,2530	30	0,0123

R: Result of the algorithm, T: Computation time (in second)

In the Leighton graphs, each graph consists of 450 vertices. First parameter of the Leighton graphs represents the number of the vertices and the second parameter represents the chromatic number [20]. Table 14 shows the results for Leighton graphs. Experimental results show that RLF algorithm finds out quite better results for Leighton graphs. Just for le450_25b graph, WP algorithm finds out the χ(G) result the better computation time. The other algorithms are generally deficient.

Table 14. The results and computation times for Leighton graphs

Graph	V	E	Den.	Eniyi/χ(G)	RLF		DSATUR		WP		LDO		IDO		FF	
					R	T	R	T	R	T	R	T	R	T	R	T
le450_15b	450	8169	0,08	15	17	0,3071	**16**	1,7589	18	0,0025	18	0,0348	18	0,9585	22	0,0337
le450_25a	450	8260	0,08	25	**25**	**0,3502**	**25**	1,7952	26	0,0029	26	0,0367	25	1,0172	28	0,0355
le450_25b	450	8263	0,08	25	25	0,3583	25	1,9924	**25**	**0,0028**	25	0,0371	25	1,0341	27	0,0355
le450_25c	450	17343	0,17	25	**28**	**0,7839**	29	5,9978	29	0,0034	29	0,0626	31	3,6658	37	0,0674
le450_5c	450	9803	0,10	5	**5**	**0,2226**	10	2,4336	12	0,0020	12	0,0352	12	1,3233	17	0,0375
le450_5d	450	9757	0,10	5	**6**	**0,2315**	12	2,4073	14	0,0025	14	0,0362	13	1,2504	18	0,0382

R: Result of the algorithm, T: Computation time (in second)

5. Conclusion

Experimental results show that while RLF and DSATUR algorithms are sufficient for the GCP, FF algorithm is generally deficient. WP algorithm finds out the best solution in the shortest time on Register Allocation, CAR, Mycielski, Stanford Miles, Book and Game graphs. On the other hand, RLF algorithm is quite better than the other algorithms on Leighton, Flat, Random (DSJC) and Stanford Queen graphs. As shown in the study, firstly it should be decided that the problems which we want solve with graph coloring algorithms is similar to what benchmark graphs. After that, the optimum graph coloring algorithms must be applied to the problem for finds out the the best solution. Thus, it can be avoided to waste of times and it can be reached the best results a quite short time.

Acknowledge

This study was supported by "Scientific Research Projects of Selcuk University". This paper has been presented as an oral presentation at the International Conference on Advanced Technology&Sciences (ICAT'16) held in KONYA (Turkey), September 01-03, 2016 and selected for the International Journal of Intelligent Systems and Applications in Engineering (IJISAE).

References

[1] D. B. West, Introduction to Graph Theory, *Prentice Hall, U. S. A., 588 pp*, 2001.

[2] J. L. Gross And J. Yellen, Graph Theory and Its Applications, CRC Press, Mathematics, 600 pages, 1998.

[3] R. Fritsch, and G. Fritsch, The Four-Color Theorem: History, Topological Foundations and Idea of Proof, Newyork: Springer, pages 260, 1998.

[4] F. Ge, Z. Wei, Y. Tian, Z. Huang, Chaotic Ant Swarm for Graph Coloring , Intelligent Computing and Intelligent Systems (ICIS), IEEE International Conference, 512-516 p., 2010.

[5] D. J. Welsh, and M. B. Powell, An upper bound for the chromatic number of a graph and its application to timetabling problems, The Computer Journal 10 (1): 85–86, 1967.

[6] I. M. Díaz and P. Zabala, A Generalization of the Graph Coloring Problem, Departamento de Computacion, Universidad de Buenes Aires, 1999.

[7] B. H. Gwee, M. H. Limand and J. S. Ho, Solving fourcolouring map problem using genetic algorithm. In Proceedings of First New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems, 332-333, 1993.

[8] N. Chmait, and K. Challita, Using Simulated Annealing and Ant-Colony Optimization Algorithms to Solve the Scheduling Problem, Computer Science and Information Technology 1(3), 208-224, 2013.

[9] K. Dowsland, J. Thompson, Ant colony optimization for the examination scheduling problem, J. Oper. Res. Soc. 56, 426–438, 2005.

[10] G.J. Chaitin, M.A. Auslander, A.K. Chandra, J. Cocke, , M.E. Hopkins and P.W. Mark-stein, Register allocation via coloring, Comput. Lang. 6 47–57, 1981.

[11] F.C. Chow, J.L. Hennessy, The priority based coloring approach to register allocation, ACM Trans. Program. Lang. Syst, 501–536, 1990.

[12] S. Ono, R. Miyamoto, S. Nakayama, and K. Mizuno, "Difficulty estimation of number place puzzle and its problem generation support." ICCAS-SICE, 2009. IEEE, 2009.

[13] W. K. Hale, Frequency assignment: *Theory and applications. Proceedings of the IEEE 12: 1497-1514*, 1980.

[14] M.R. Garey and D.S. Johnson, Computers and intractability, in: A Guide to the Theoryof NP-Completeness, W.H. Freeman & Co., New York, NY, U.S.A., 1979.

[15] S. Mahmoudi and S. Lotfi, Modified cuckoo optimization algorithm (MCOA) to solve graph coloring problem, *Applied soft computing Volume 33, 48–64*, 2015.

[16] A. Hertz and D. de Werra, Using tabu search techniques for graph coloring, *Computing 39, 345–351*, 1987.

[17] M. Chams, A. Hertz and D. de Werra, Some experiments with simulated annealing for coloring graphs, *European Journal of Operational Research, 32(2):260-266*, 1987.

[18] S. Ahn, S. Lee, T. Chung, Modified Ant Colony System for Coloring Graphs, *Information, Communications and Signal Processing, 2003 and Fourth Pacific Rim Conference on Multimedia. Proceedings of the Joint Conference of the Fourth International Conference on, IEEE, 1849 – 1853*, 2003.

[19] H. Al-Omari and K.E. Sabri, New Graph Coloring Algorithms, *American Journal of Mathematics and Statistics 2 (4): 739-741*, 2006.

[20] F.T. Leighton, A graph coloring algorithm for large scheduling problems, *J. Res.Nat. Bur. Stand. 84 489–506*, 1979.

[21] D. Brélaz, New methods to color the vertices of a graph, *Commun. ACM 22 251–256*, 1979.

[22] DIMACS graph coloring instances. (2016) instances homepage on CMU. [online]. Available: http:mat.gsia.cmu.edu/COLOR/instances.html.

[23] I. C. R. Ruiz, "Gravitational Swarm for Graph Coloring", PHD thesis, The University of the Basque Country Donostia - San Sebastian, 2012.

[24] D. S. Johnson and M. A. Trick, Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, 1993, vol. 26 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society. 1996.

[25] M. Caramia and P. Dell'Olmo. A lower bound on the chromatic number of mycielski graphs. Discrete Mathematics, 235(1-3):79_86, 2001.

[26] B. Yılmaz, A novel meta-heuristic for graph coloring problem: simulated annealing with backtracking, Yeditepe University Graduate School Of Natural Scıences, Istanbul, 2011.