IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Revue d'Intelligence Artificielle
Vol. 33, No. 3, June, 2019, pp. 189-195
Journal homepage: http://iieta.org/journals/ria

A Linked List-Based Exact Algorithm for Graph Coloring Problem

Ajay Narayan Shukla*, Vishal Bharti, Madan L. Garag

DIT University, Dehradun 248001, India

Corresponding Author Email: ajay.narayan.shukla@dituniversity.edu.in

https://doi.org/10.18280/ria.330304

Received: 22 March 2019
Accepted: 7 June 2019

Keywords:
graph coloring, adjacency matrix, singly linked list, undirected graph

ABSTRACT

Graph coloring is an NP-hard problem. There is ample room to further reduce the number of colors being used under the strict constraints of the problem. This paper proposes an algorithm that stores the information of the vertices in the graph, together with the color and the next address field of a node in a singly linked list. The coloring of a particular vertex is decided on the content of color field of node corresponding to the searched vertex for coloring. The adjacency matrix of the graph was adopted to select the feasible color to be allocated to a vertex, without violating the constraints for coloring of the vertices in the graph. The proposed algorithm was tested on DIMACS benchmark graphs, using Python 3.6. The results show that our algorithm produced the result with the optimal number of colors required to solve the problem. This research provides a new strategy to find the optimal coloring solution for each type of graphs.

1. INTRODUCTION

Graph coloring problem is most studied combinatorial problem in graph theory. Graph coloring explored in literature for solution due wide verity of its uses in many real world engineering applications. Given an undirected graph G=(V,E) where V is set of vertices & E is set of edges in the graph G, the coloring problem involve to find the distinct colors that can use in assigning the colors to the vertices in the graph. The number of distinct colors used to color the graph should be minimum with the constraints such that no adjacent vertices in graph have assigned same color. The graph colored with minimum number of distinct colors is also referred as the chromatic number of the graph.

The coloring problem of graph is itself a NP-class problem and therefore researchers tried to find the solution of the problem by adopting some new techniques compared to previous existing techniques for their proposed algorithms so that it running time may be more optimized. There are mostly two categories of algorithm exists in literature to solve the problem: approximate algorithms and exact algorithms. The applications where graph coloring has used to find the efficient solution are satellite range scheduling problem [1]. The communication scheduling problem between satellite and ground station was addressed by using the heuristics of graph coloring problem. The researchers have addressed the problem for allocation of variables in a loop to the available registers in computer by encoding the problem instance into graph coloring problem [2]. The researchers have also addressed the a-graph coloring problem for near-triangulation of the plane with a face size of 4 and develop the coloring condition by transforming it to a 4-color problem [3]. Since the coloring of the vertices in the graph is NP-class problem and therefore one aspect of the problem is also related to complexity of the algorithms available in literature. The issue of complexity for special classes of graphs related to coloring of vertices in the graph also discussed by the researchers and well reported [4]. Usefulness of this problem has discussed in parallel computing environment [5] and researchers gave their new approach in the form of balanced coloring models. The problem of coloring is addressed by the researchers for partition graph coloring [6] and gave the solution for the problem based on hybrid ant-local search technique that involves the partitioning of the given graph into number of different sets. The applications of graph coloring problem are such a versatile in nature that researchers have always tried to solve the problem using techniques that may suitable to apply the solution for a particular application. The different heuristics [7] has analyzed and discussed for the working of the procedures to get the solution of the problem. The time table scheduling for exam conduction in University using graph coloring is also reported [8]. Authors addressed the constraints for time table scheduling of examination and proposed the solution for the problem using cluster heuristics & sequential heuristic. Several more applications of graph coloring problem have reported in literature [9-11]. The problem of simultaneous utilization of base antennas for multi cell multiple–input multiple-output system is addressed by the researchers [12] and proposed the solution in the form of weighted graph coloring.

In this research work we propose the algorithm which uses singly link list to store the information of the vertices of the graph along with color and next address field of a node in the created link list. The coloring of a particular vertex is decided on the content of color field of node corresponding to the searched vertex for coloring. The proposed procedure uses adjacency matrix for the graph that is used to make a decision for feasible color that may be allocated to a vertex without violating the constraints for coloring of the vertices in the graph.

The proposed procedure is always terminated by assigning appropriate color to each vertex in the graph and since it never generated other implicit constraints during coloring process therefore the running time is efficient. This algorithm can be

used for solving any real world application that involves the use of graph coloring problem without affecting its complexity.

2. RELEATED WORK

The algorithms proposed by various researchers may generally falls into two categories: exact algorithms & approximate algorithms. The exact algorithms include the solution of graph coloring problem based on integer programing formulation [13], technique based on linear decomposition of graph [14], branch and cut algorithms [15], using set covering formulation of the given graph [16] and based on DSATUR algorithm [17].

The other class of algorithms includes heuristics based greedy subroutines [18], hybrid evolutionary algorithm [19], by strengthening the construction in existing ant colony optimization [20] and coloring of vertices in the graph using amalgamation of local search & constraints programming [21].

The algorithm based on clause learning is reported [22], in which the authors gave the solution for graph coloring problem and they used propositional logic to search implicit constraints generated in the process of coloring of the vertices in the given graph. The solution for coloring problem has also given by the researchers using the concept of DNA computing [23]. In their proposed approach authors have developed a parallel type DNA computing model based on working of polymerase chain reaction (PCR). The coloring problem of graph is also addressed and given the efficient solution using list data structure by the researchers [24-25]. The researchers have also given the solution of the problem by adopting the approach based on adjacency matrix to evolve the solution for the problem, which is applicable to any kind of graph [26]. The brief description of their proposed algorithms are as follows:

Algorithm 1

Step 1- Select any one vertex in the given graph and assign the color from the information obtained from color adjacency matrix.

Step 2- Update color adjacency matrix for those vertices which are adjacent vertex to the vertex that has assigned a color in previous step.

Step 3- Repeat step 1 & step 2 until all the vertices of the graph have explored & assigned some color.

The researchers have created an additional color matrix with dimension of nX k, where n is number of vertices in the graph & k is the number of available colors. Initially color matrix is initialized with value 1.

Assigning of the color to a particular vertex is done based on content of color matrix and then this matrix is searched for those non-adjacent vertices for which assigned color to current vertex may be act as one of feasible color otherwise this feasible may also be updated.

In another approach researchers addressed the solution for the graph coloring problem and they proposed a new technique by creating binary search tree and then with created binary search tree, authors constructed the heap [27]. The authors created a new data structure in which the available colors for the graph represented in the form of a collection of binary search tree. The root of each binary search tree constructed contains information about single vertex and rest nodes of a tree contains feasible colors. So in case if there is n number of vertices in any graph that can colored with k number of different colors, there proposed technique uses n number of binary trees and in each tree contains k+1 number of nodes initially. Entire collection of these binary search trees grouped in the form heap. The procedure used to solve the problem is as under:

Algorithm 2

Step 1: select a vertex from the graph (this selection has done through adjacency matrix of graph).

Step 2: find the suitable color from the heap of binary search tree and assign the color to selected vertex. It is children of root node of that binary search tree whose root contains the vertex information.

Step 3: update the heap by removing assigned color node from those binary search trees whose root node contains the vertex that is adjacent vertex to currently assigned color vertex.

Step 4: repeat from step 1 to step 3 until all vertices in the graph have assigned some color.

Although the above algorithm works satisfactorily and always terminated with coloring solution but in step 2 searching for appropriate color to any vertex involves the traversal of the unexplored heap of previously colored vertex. This additional cost of traversal through colored nodes may be avoided.

Several other methodologies have also find in literature to solve the problem of coloring of graphs that involves: improved cuckoo optimization technique [28], using coupled oscillatory networks [29], using local anti-magic labeling in the graph [30], cellular learning automata based solution [31], solving graph-b coloring problem using hybrid genetic algorithms [32], Douglas-Rachford Algorithm based solution [33] for graph coloring problem and branch-and-cut algorithm for the equitable coloring problem [34].

The graph coloring problem involved in diversified area of applications, which always enables the researchers to give the solution for problem in a manner that can applied efficiently to a particular application.

3. MOTIVATION FOR OUR APPROACH

Graph coloring problem is probably one of the NP-class problems mostly studied for its various applications and solution of the problem involves different techniques suggested by the researchers. There are various optimization techniques available in the literature applied to the problem for getting the solution. Few more techniques also find in literature that uses traditional data structures to find the solution. Sometimes one technique use to color the vertex in the graph may not found suitable for some classes of graphs that may generated for a particular application and secondly since this problem is a NP-class problem so there is always possibility to evolve a new procedure for coloring of the vertices in the with more optimize running time.

In the proposed work we tried to develop a procedure that works for any kind of graph and therefore it may be apply to solve any application problem involving graph coloring technique.

4. DATA STRUCTURE USED FOR THE ALGORITHM

The propose algorithm uses adjacency matrix of corresponding graph along with linear link list to solve the

problem. Suppose a graph given in Figure 1 which is a two-colorable undirected graph having five vertices and five number of edges.

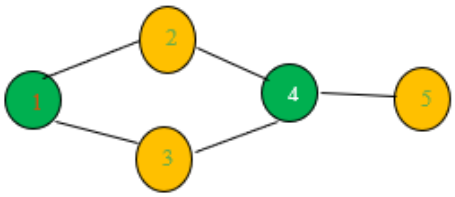

Figure 1. A 2-colorable graph

The matrix required to represent the above given graph is 5X5 square adjacency matrix in which matrx(i,j) =1 if there is a path between vertex i and j , otherwise the entry in the matrix corresponding to matrix(i, j)=0. Therefore, for a graph with n number of vertices required a two dimensional array of size n*n to store graph information.

The matrix generated in the form of adjacency matrix for the graph in Figure 1 is shown in Table 1. The content of matrix will use to create the set of feasible colors on the basis of created link list that requires the knowledge of adjacent vertices of a vertex that has assigned some color.

Table 1. Adjacency matrix representation of Figure 1

	1	**2**	**3**	**4**	**5**
1	0	1	1	0	0
2	1	0	0	1	0
3	1	0	0	1	0
4	0	1	1	0	1
5	0	0	0	1	0

The objective of liner link list in the procedure is twofold: in first phase the color field of a particular node gives the actual color value for the vertex field of node, which will finally store in a one dimensional array created in procedure. Once color information fetched from node in the list for current vertex, the list will be updated in color field for those nodes which are adjacent to currently assigned color vertex. The general structure of nodes in the link list used in algorithm is shown in Figure 2.

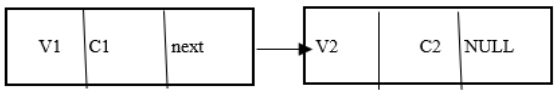

Figure 2. Proposed structure of link list used in algorithm

In the above proposed structure of link list the first field of a node will store the vertex information of the graph numbered from 0 to n-1, if graph is having n number of vertices. The second field is use to color information numbered from 0 to k-1 for a k-colorable graph. Initially the list has created with vertex information for all the vertices in the graph along with color field of all nodes in the list as 0. Therefore, the list contains n number of nodes for a graph having n number of vertices.

5. PROPOSED ALGORITHMS

The methodology for design of propose algorithms consists of adjacency matrix A[ ][ ] for graph, a singly link list as represented in Figure 2 the fields of each node contains vertex in the form of numeric value, color information in the form of integral values and next address containing address of adjacent node. The next address field of last node in the list will be NULL. Initially the list has been created by filling the vertex number, color value is 0 for all the nodes in list. The final color information corresponding to each vertex is stored in a one dimensional array C[ ] which has created with size of number of vertices in the graph.

Algorithms 3 assign_color(n)

```
{
Create adjacency matrix A[ ][ ] for the graph;
Create a list with n nodes;
Let first node of the list is pointed with start pointer;
 for i= 0 to n-1
 {
  K=C[i]= start-> color
  Delete node( start,i);
   update _list(start, A, i, k);
 }
}
```

In the above algorithm, initially a list of n nodes has created where n is the number of vertices in the graph. The color information corresponding to vertex i is stored in array C along with in another variable k. After retrieving the color from the node containing vertex i, node has deleted from the list. Then finally update_list procedure has called to update the colors in all adjacent nodes to vertex i.

Algorithm 4 update_list(start, A, i, k)

```
{
for j= i+1 upto  n-1
{
 if(A[i][j]= = 1)
 {
   p= start;
   while( p->information != j and p != NULL)
      p= p->next;
  if( p->color= = k)
     p->color= k+1;
  }
 }
}
```

The above procedure involves the updating of color field of those nodes in the list which contains adjacent vertices information of vertex i. The adjacent vertices of currently assigned color vertex can be find the basis content of the adjacency matrix A. If any vertex j is adjacent vertex to vertex i then A[i][j] will be 1. Once adjacent vertex has found, the previous color information available in the color field of corresponding node in the list updated with new feasible color value k+1.

Initially assign_color ( ) procedure involves the creation of adjacency matrix and link with described fields. After the creation of list, the algorithm assigns the color for a particular vertex i by fetching the color information from the node of the list containing vertex information i and then it invoke update_list( ) procedure for updating the color field of adjacent vertices of currently colored vertex.

5.1 Analysis of the proposed algorithms

The proposed algorithm started with creation of creation of liner link list containing n number of nodes along with the creation of an adjacency square matrix of order nXn. The running time analysis of proposed algorithms involved to find the maximum number of times instructions executed in proposed algorithm 3 & algorithms 4.

Suppose the graph on which the presented algorithms applied for coloring consists of n number of vertices and the number of colors used to color the graph is k.

Let the number of vertices in the given graph is n and the minimum number of available colors that can use to color the vertices of the graph is k. The execution time of the presented procedures can be computed formally using following computational technique available in algorithms.

(1). The list creation executed in O(n) times in assign_color ( ) procedure.

(2). The adjacency matrix formation required either O(n2) time or it may involve by reading the files from disk which is one time.

(3). The procedure assign_color( ) uses to compute the actual color value for a vertex and it requires O(n) running time.

(4). delete_node( ) procedure will executed with constant running time.

(5). update_list ( ) procedure invoked n times and the instructions for finding the adjacent vertices for currently assigned color node and updating the corresponding color field in the list will require maximum $O(n^2)$ execution time. Although in every step it will be reduced.

(6). Since the update_list( ) invoked n times so the total running time for this procedure will be $O(n^3)$.

After the analysis of all the above steps the estimated execution time by leaving the additive less significant terms for the algorithms proposed will be $O(n^3)$.

6. EXPERIMENTAL RESULTS AND DISCUSSION

The proposed algorithms in section 5 is tested on few simple graphs generated by us initially for its working. Generally, graph coloring algorithms available in literature tested against DIMACS graph coloring instances. We also tested the proposed algorithms on several graph coloring instances of DIMACS and find that in most of the cases where the graph is a kind of undirected graph, our algorithm will always comparable with the best known results.

We used Python 3.6 that reads the DIMACS graph coloring instances and converted the graph into equivalent adjacency matrix stored in the text file. The graphs that were used to test the algorithms proposed, taken from DIMACS benchmark graphs from the source [35]. This source contains different class of graphs in the form of vertex-edge list. Several graphs were taken from the above source for testing for which the description is given in the Table 2.

All above files is in form of (V, E) for a particular file. For example, myciel3.col (11, 20) file consists of eleven vertices and twenty edges in the graph. We have used Python that read the files in the above format and converted the file in the form of adjacency matrix, which stored in a text file. The converted text file for myciel4.col (23, 71) is shown in the form of adjacency matrix in Table 3.

Table 2. Test cases from source [35]

Graph Name	Description
myciel3.col (11,20)	Triangle free graphs based on Mycielski transformation which cannot solve easily with property that coloring increase in problem size.
myciel4.col (23,71)	
myciel5.col (47,236)	
myciel6.col (95,755)	
myciel7.col (191,2360)	
2-Insertions_3.col (37,720	Generalization of myciel graphs by adding more vertices for increasing the graph size but at the same without changing the density of graph.
3-Insertions_3.col (56,110)	
queen5_5.col (25,320)	For the given graph the maximum clique is not more than 5
david.col (87,812)	Graphs translated from Stanford graphbase files
huck.col (74,602)	
jean.col (80,508)	
mugg88 1.col (88,146)	Very hard 3-coloribilty instances from Kazunori Mizuno.
mugg88 25.col (88,146)	
4-Insertions_3.col (79,156)	4-Insertions graph of order 3%
1-Insertions_4.col (67,232)	1-Insertions graph of order 4%
1-FullIns_3.col (30,100)	1-FullIns graph of order 3
2-FullIns_3.col (52,201)	2-FullIns graph of order 3

Table 3. Adjacency matrix for myciel4.col (23, 71) graph

```
0 1 0 1 0 0 1 0 1 0 0 0 1 0 1 0 0 1 0 1 0 0 0
0 0 1 0 0 1 0 1 0 0 0 1 0 1 0 0 1 0 1 0 0 0 0
0 0 0 0 1 0 1 0 0 1 0 0 1 0 0 1 0 1 0 0 1 0 0
0 0 0 0 1 1 0 0 0 1 0 1 0 0 0 1 1 0 0 0 1 0 0
0 0 0 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

We implemented the algorithm in C language for the test cases of DIMACS graph coloring instances. After converting the graph into text file we used retrieved the adjacency matrix from text file and stored the content of the matrix into a two dimensional array a[ ][ ]. The command used for retrieving the content of text file is:

FILE *in = fopen("myciel4.txt", "r");

In the above statement "in" is s file pointer and "myciel4m.txt" is the name of the file and the file opened in read mode for getting the content from the file. We created a link list corresponding to each graph with fields as shown in figure 2 with n number of nodes where n is number of vertices in the selected graph for testing. The vertex information stored in the form integer values varied from 0 to n-1. Similarly, color value for each node initialized with 0. The result obtained after running the proposed algorithms for the test cases given in Table 2 and its comparison with existing techniques for same set graph coloring instances is shown in Table 4. We used Turbo C3.2.2.0 compiler over windows7 platform (32 bit) with hardware configuration Intel Pentium D CPU clock speed 2.8 GHZ and 4 GB RAM.

We compared with MCOA technique [28] to solve the graph coloring problem and found that for most of the above mentioned test cases our algorithm works satisfactorily and produced optimal coloring solution for the tested graphs.

The table below contains the results obtained after implementing the proposed algorithms in terms of number of colors required to color a particular graph. The first column of the table contains the name of graph along with number of vertices and number of edges in the graph. The values in the second column in the table is best known number of colors corresponding to graph in the first column. The entry in the third column is our result that achieved after running the proposed algorithms on the above given data set. Finlay the last column entries are the results of technique used in Modified cuckoo optimization algorithm [28].

The number of colors for myciel graphs myciel3, myciel4, myciel5, myciel6, myciel7, based on Mycielski transformation is same in our case to best known for all the above compared graphs. It is also similar and comparable with the techniques used in [28] for graph coloring.

Similarly, we tested graph queen5_5, david, huck, jean, from Stanford GraphBase File and get result which is equal to best known color value for the graph. The results after running the proposed algorithm for the graphs mugg88.1 and mugg88.25 graphs from Kuzunori Mizuno, the number of colors required to color above graphs and obtained number of color values with our algorithm is same.

When we tested the graph 1-Insertions-4, graph of order 4 % our algorithm uses to five colors to color the entire graph however the best known optimal color value for this graph is four. But our result is similar to result in the case of MOCA technique. After testing the graph 4-Insertions-3, graph of order 3 %, our algorithm requires four colors to color the whole graph for which the best optimal color value is three. However, our result is similar to the result obtained in case of MOCA technique applied to solve the problem.

When we run our proposed algorithm on the 1-FullIns_3 graph of order 3 and 2-FullIns_ 3 graph of order 3 found that the number of colors used for the coloring of graph 1-Fullins_3 is eight and for the graph 2-Fullins3 it was ten. However, in case of MOCA [28] technique for graph coloring for these graphs, the optimal colors used is four colors and five colors, respectively.

Table 4. Comparison of proposed algorithms with MOCA [28]

Graph Name	Best Known [35]	Our Case	MOCA [28]
myciel3.col (11,20)	4	**4**	4
myciel4.col (23,71)	5	**5**	5
myciel5.col (47,236)	6	**6**	6
myciel6.col (95,755)	7	**7**	7
myciel7.col (191,2360)	8	**8**	8
2-Insertions_3.col (37,720	4	**4**	4
3-Insertions_3.col (56,110)	4	**4**	4
queen5_5.col (25,320)	5	**5**	5
david.col (87,812)	11	**11**	11
huck.col (74,602)	11	**11**	11
jean.col (80,508)	10	**10**	10
mugg88 1.col (88,146)	4	**4**	4
mugg88 25.col (88,146)	4	**4**	4
4-Insertions_3.col (79,156)	3	4	4
1-Insertions_4.col (67,232)	4	5	5
1-FullIns_3.col (30,100)	-	8	4
2-FullIns_3.col (52,201)	-	10	5

Probably for last four cases our algorithms terminated with considerable number of more colors as compared to existing one is due to approach that we are using to decide a color for a particular vertex in the graph. In our algorithm when a color is given to a vertex, all the adjacent vertices of that that node assigned temporarily one higher integral color value. Due to this there may be the vertices which are non-adjacent vertex to previous assigned colored vertex but since the color field for current vertex is already having different than its non-adjacent vertices. Therefore, the feasible color value for current vertex remains unchanged and thus it increases coloring values for successive vertices and their corresponding adjacent verities in the graph.

In spite of except for last two graphs our algorithms work efficiently and successfully find either best color values for the graphs or comparable with existing known color values. We used existing data structure link list and array for finding the solution of the problem by focusing only constraint associated with graph coloring problem. The proposed algorithm is efficient in its working because we removing every vertex from the list once it assigned a particular and therefore next vertex for coloring is always available as the first node in the list. Similarly, at the time of searching adjacent vertices for

currently assigned color and modifications of their color values is also be done in optimize manner.

7. CONCLUSION AND FUTURE SCOPE

The proposed research work intends to devise a new approach to address the graph coloring problem that must able to find the exact solution for the problem. In the proposed approach we tried to give the exact solution for graph coloring problem. The proposed technique uses the link list to vertex information for coloring and adjacency matrix for graph representation. We used mentioned data structures for finding the solution because these data structures are simple and efficient in terms of required operation that performed in find the solution of the problem. Although for few cases the proposed algorithms fail work efficiently but there is no any single technique in the literature which able to find the optimal solution for each kind of graphs. The best known solutions for the different category of graphs are the result of several techniques. The proposed approach will work for any kind of graphs however for some cases like discussed in previous section if graph falls in that mentioned category may not works satisfactorily.

As the graph coloring problem is a NP-class problem so there is always hope for the optimizing the time complexity of the algorithm for solving the problem. In our proposed algorithms after incorporating some techniques for selecting the feasible colors among the previously assigned colors with minimum colour index may able to produce better results for those class of graphs which still needs to find optimal coloring solution.

REFERENCES

[1] Zufferey, N., Amstutz, P., Giaccari, P. (2008). Graph coloring approaches for a satellite range scheduling problem. Journal of Scheduling, 11(4): 263-277. https://doi.org/10.1007/s10951-008-0066-8

[2] de Werra, D., Eisenbeis, C., Lelait, S., Marmol, B. (1999). On a graph theoretical model for cyclic register allocation. Discrete Applied Mathematics, 93(2-3): 191-203. https://doi.org/10.1016/S0166-218X(99)00105-5

[3] Tilley, J.A. (2017). The *a*-graph coloring problem. Discrete Applied Mathematics, 217(2): 304-317. https://doi.org/10.1016/j.dam.2016.09.011

[4] Demange, M., Monnot, J., Pop, P., Ries, B. (2014). On the complexity of the selective graph coloring problem in some special classes of graphs. Theoretical Computer Science, 540-541: 89-102. https://doi.org/10.1016/j.tcs.2013.04.018

[5] Lu, H., Halappanavar, M., Chavarría-Miranda, D., Gebremedhin, A.H., Panyala A., Kalyanaraman, A. (2017). Algorithms for balanced graph colorings with applications in parallel computing. IEEE Transactions on Parallel and Distributed Systems, 28(5): 1240-1256. https://doi.org/10.1109/TPDS.2016.2620142

[6] Fidanova, S., Pop, P. (2016). An improved hybrid ant-local search algorithm for the partition graph coloring problem. Journal of Computational and Applied Mathematics, 293: 55-61. https://doi.org/10.1016/j.cam.2015.04.030

[7] Galinier, P., Hamiez, J.P., Hao, J.K., Porumbel, D. (2013). Recent advances in graph vertex coloring. In: Zelinka I., Snášel V., Abraham A. (eds) Handbook of Optimization. Intelligent Systems Reference Library, vol 38. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-30504-7_20

[8] Hussin, B., Basari, A.S.H., Shibghatullah, A.S., Asmai, S.A., Othman, N.S. (2011). Exam timetabling using graph colouring approach. 2011 IEEE Conference on Open Systems, Langkawi, pp. 133-138. https://doi.org/10.1109/ICOS.2011.6079274

[9] Gaceb, D., Eglin, V., Lebourgeois, F., Emptoz, H. (2009). Robust approach of address block localization in business mail by graph coloring. The International Arab Journal of Information Technology, 6(3): 221-230.

[10] Khasawneh, M.A., Malkawi, M.I., Hayajneh, T.S. (2009). A graph-coloring-based navigational algorithm for personnel safety in nuclear applications. 2009 6th International Symposium on Mechatronics and its Applications, Sharjah, pp. 1-7. https://doi.org/10.1109/ISMA.2009.5164808

[11] Selemani, M.A, Mujuni, E., Mushi, A. (2013). An examination scheduling algorithm using graph colouring- the case of Sokoine University of agriculture. International Journal of Computer Engineering & Applications, II(I/III): 116-127.

[12] Zhu, X.D., Dai, L.L., Wang, Z.C., Wang, X.D. (2017). Weighted-graph-coloring-based pilot decontamination for multicell massive MIMO systems. IEEE Transactions on Vehicular Technology, 66(3): 2829-2834. https://doi.org/10.1109/TVT.2016.2572203

[13] Méndez Díaz, I., Zabala, P. (2008). A cutting plane algorithm for graph coloring. Discrete Applied Mathematics, 156(2): 159-179. https://doi.org/10.1016/j.dam.2006.07.010

[14] Lucet, C., Mendes, F., Moukrim, A. (2006). An exact method for graph coloring. Computers & Operations Research, 33(8): 2189-207. https://doi.org/10.1016/j.cor.2005.01.008

[15] Méndez-Díaz, I., Zabala, P. (2006). A branch-and cut algorithm for graph coloring. Discrete Applied Mathematics, 154(5): 826-847. https://doi.org/10.1016/j.dam.2005.05.022

[16] Malaguti, E., Monaci, M., Toth, P. (2011). An exact approach for the Vertex Coloring Problem. Discrete Optimization 8(2): 174-190. https://doi.org/10.1016/j.disopt.2010.07.005

[17] Segundo, P.S. (2012). A new DSATUR-based algorithm for exact vertex coloring. Computer & Operations Research, 39(7): 1724-1733. https://doi.org/10.1016/j.cor.2011.10.008

[18] Caramia, M., Dell'Olmo, P. (2008). Coloring graphs by iterated local search traversing feasible and infeasible solutions. Discrete Applied Mathematics, 156(2): 201-217. https://doi.org/10.1016/j.dam.2006.07.013

[19] Porumbel, D.C., Hao, J.K., Kuntz, P. (2009). Diversity control and multi parent recombination for evolutionary graph coloring algorithms. 9th European Conference on Evolutionary Computation in Combinatorial Optimization (EVOCOP2009). Tbingen, Germany, pp. 121-132. https://doi.org/10.1007/978-3-642-01009-5_11

[20] Dowsland, K.A., Thompson, J.M. (2008). An improved ant colony optimization heuristic for graph coloring Discrete Applied Mathematics, 156(3): 313-324. https://doi.org/10.1016/j.dam.2007.03.025

[21] Prestwich, S.D. (2008). Generalized graph coloring by a hybrid of local search and constraint programming. Discrete Applied Mathematics, 156(2): 148-158. https://doi.org/10.1016/j.dam.2006.07.011
[22] Zhou, Z.Y., Li, C.M., Huang, C., Xu, R.C. (2014). An exact algorithm with learning for the graph coloring problem. Computers & Operation Research, 51: 282-301. https://doi.org/10.1016/j.cor.2014.05.017
[23] Xu, J., Qiang, X.L., Zhang, K., Zhang, C., Yang, J. (2018). A DNA computing model for the graph vertex coloring problem based on probe graph. Engineering, 4(1): 61-77. https://doi.org/10.1016/j.eng.2018.02.011
[24] Shukla, A.N., Garg, M.L., Misra, R. (2019). An approach to solve graph coloring problem using linked list. International Journal of Advanced Studies of Scientific Research, 4(2).
[25] Shukl, A.N., Garg, M.L. (2018). A list based approach to solve graph coloring problem. 2018 International Conference on System Modeling & Advancement in Research Trends (SMART), Moradabad, India, pp. 265-267. https://doi.org/10.1109/SYSMART.2018.8746966
[26] Shukla, A.N., Garg, M.L. (2019). An approach to solve graph coloring problem using adjacency matrix. Oryzae. Biosc. Biotech. Res. Comm., 12(2). http://dx.doi.org/10.21786/bbrc/12.2/33
[27] Shukla, A.N., Bharti, V., Garg, M.L. (2019). An algorithm based on heap of binary search tree to solve graph coloring problem. International Journal of Recent Technology and Engineering (IJRTE), 8(2): 3920-3924.
[28] Mahmoudi, S., Lotfi, S. (2015). Modified cuckoo optimization algorithm (MCOA) to solve graph coloring problem. Applied Soft Computing, 33: 48-64. https://doi.org/10.1016/j.asoc.2015.04.020
[29] Parihar, A., Shukla, N., Jerry, M., Datta, S., Raychowdhury, A. (2017). Vertex coloring of graphs via phase dynamics of coupled oscillatory networks. Scientific Reports, 7: 11. https://doi.org/10.1038/s41598-017-00825-1
[30] Arumugam, S., Premalatha, K., Bača, M., Semaničová-Feňovčíková, A. (2017). Local antimagic vertex coloring of a graph. Graphs and Combinatorics, 33(2): 275-285. https://doi.org/10.1007/s00373-017-1758-7
[31] Rezapoor Mirsaleh, M., Meybodi, M.R. (2016). A new memetic algorithm based on cellular learning automata for solving the vertex coloring problem. Memetic Computing, 8(3): 211-222. https://doi.org/10.1007/s12293-016-0183-4
[32] Labed, S., Kout, A., Chikhi, S. (2018). Solving the graph b-coloring problem with hybrid genetic algorithm. 2018 3rd International Conference on Pattern Analysis and Intelligent Systems (PAIS), Tebessa, pp. 1-7. https://doi.org/10.1109/PAIS.2018.8598525
[33] Aragón Artacho, F.J., Campoy, R. (2018). Solving graph coloring problems with the Douglas-Rachford algorithm. Set-Valued and Variational Analysis, 26(2): 277-304. https://doi.org/10.1007/s11228-017-0461-4
[34] Bahiense, L., Frota, Y., Noronha, T.F., Ribeiro, C.C. (2014). A branch-and-cut algorithm for the equitable coloring problem using a Formulation by representatives. Discrete Applied Mathematics, 164: 34-46. https://doi.org/10.1016/j.dam.2011.10.008
[35] https://turing.cs.hbg.psu.edu/txn131/graphcoloring.html #XXCAR, accessed on 10 March 2019.