Indian Journal of Science and Technology, *Vol 10(25), DOI: 10.17485/ijst/2017/v10i25/115117, July 2017*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Parallel Computing Aspects in Improved Edge Cover Based Graph Coloring Algorithm

Harish Patidar*, Prasun Chakrabarti and Amrit Ghosh

Department of Computer Science and Engineering, Sir Padampat Singhania University, Udaipur – 313601, Rajasthan, India; harish.patidar@gmail.com

Abstract

Objective: To improve the Edge Cover based Graph Coloring Algorithm (ECGCA) using independent set by incorporating parallel computing aspects in algorithm. Finding optimum time complexity is one of the main objectives of this paper. **Methods/Statistical Analysis:** This paper introduced some modification in ECGCA. Algorithm is implemented and tested using Java programming language. Java multithreading concept is used to achieve parallel computing in algorithm. DIMACS graph instances are used to test algorithm. **Finding:** Algorithm is tested on more than 75 DIMACS graph instances. To analyze the time complexity, execution time of algorithm in seconds is calculated by program. Algorithm is tested on different DIMACS graph instances. Test data is analyze in this paper and found that proposed algorithm executed in optimum time for large graphs. This paper also compared parallel algorithm and found that proposed parallel algorithm is less time complex than sequential algorithm. Most of the exact graph coloring algorithms are not suitable for large graph (more than 100 vertices) but proposed algorithm is tested on many large graphs and high execution success rate of algorithm is achieved. **Application:** This paper shows the experimental results of different type of application data. It means this algorithm can be used for maximum types of applications.

Keywords: Edge Cover, Graph Coloring, Independent Set, Multithreading, Parallel Computing, Vertex Coloring

1. Introduction

Graph coloring problem is well known NP hard problem.[1,2] Graph coloring problem can be used to solve many real word applications. Graph coloring problem has three different areas of applications.

First, vertex coloring i.e. all the vertices of any graph must be colored in such a way that no two connected vertices colored with the same color. And number of color required to color the vertices should be optimum. This number is also known as chromatic number. There is huge area of applications of vertices coloring like, register allocation in operating system, frequency assignment, scheduling problem solving, time table scheduling, puzzles solving etc.

Second, edge coloring i.e. all edges of graph must be colored in such a way that any edges connected with a single vertex must not have the same color. The applications of edge coloring are path coloring in fiber optic communication, time division multiple access network communication, open shop scheduling, round robin tournament etc.

Third, face coloring i.e. coloring the faces of graphs in such a manner that faces with common boundary must not be colored with the same color. Mainly face coloring is the origin of graph coloring problem. Primarily face coloring is used to solve the map problem, in which geographical map is designed with four colors. Among all this three area this paper proposed an algorithm which can be used to solve vertex coloring problem.

As the application area of graph coloring is increasing and size of problem data is also increasing. Graph coloring algorithm can be used in designing in flexible manufacturing system, multi-beam satellite system by making optimum use of beams, time table scheduling, and Air traffic flow management.[3-5]

**Author for correspondence*

Graph coloring is used by many internet applications, and data size in internet applications is very large like social networking sites. There is big challenge for researchers to provide algorithms which can give results in optimum time. All though researcher primary goal is to find optimum chromatic number, but when graph coloring algorithm is developed for large graphs, execution time is more important than chromatic number. This paper proposed some improvement in edge cover based graph coloring algorithm to optimize execution time for large graphs. This algorithm is based on finding maximal independent set using edge cover technique.

There are many approaches to solve graph coloring problems like approximate solution through greedy constructive, Meta heuristics and local search heuristics. There are some exact solutions approaches are also available. Solving graph coloring problem through maximal independent set is one of them.

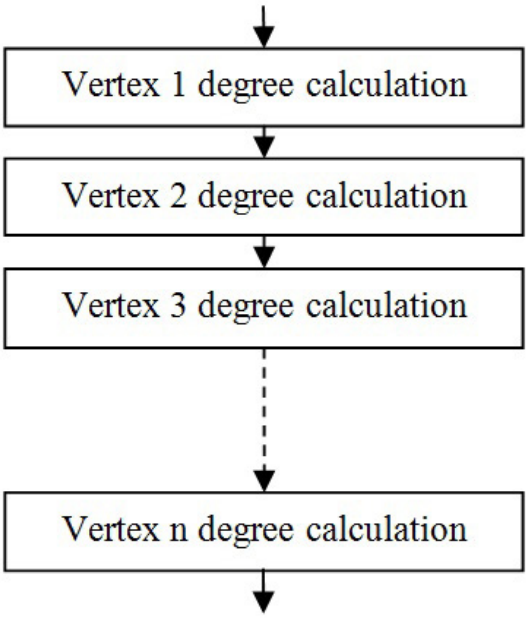

Figure 1. Degree calculation step of ECGCA.

It has proposed an algorithm to find an independent set in $O(2^{0.276n})$ time.[6] Algorithm is based on three main sources first, a modified recursive algorithm. Second, an improvement in time bound. And third, a time space trade off. A simple parallel algorithm is proposed to find maximum independent set.[7] It has proposed an approximate algorithm for independent set in 3 uniform hyper graphs for the coloring problem.[8] It has proposed a polynomial time algorithm for finding independent set.[9] A general dynamic programming algorithm for finding the maximum weight independent set problem is developed.[10] In 2015 Brændeland introduced a greedy algorithm for finding maximum independent set.[11] This algorithm is iteration based algorithm iterates on uniformly sized independent sets. It has proposed an algorithm, which overcome the processing overhead of Kernelization techniques. Kernelization technique is used to find out the exact maximum independent sets.[12] To solve the graph coloring problem finding independent set problem can be used as an effective solution.

Rest of the paper is organized as follows: In Section 2, Edge cover based graphs coloring algorithm concepts are discussed. In Section 3, proposed improvement in algorithms are included. In Section 4, Experimental results on datasets are shown. In Section 5, Experimental results analysis is included. In Section 6, effect of multithreading on algorithm is discussed. And in Section 7, paper is concluded and some future scopes are also discussed.

2. Edge Cover Based Graph Coloring Algorithm (ECGCA)

ECGCA finds independent set using edge cover technique. Edge cover technique selects the minimum vertices such that all edges of graph are covered and reaming vertices are included into independent set.

Following steps followed by ECGCA:

Step 1, Create edge set, vertex set and an empty edge cover set.

Step 2, Find independent set.

This step is core of algorithm and this step has three sub steps:

i. Create temporary copies of vertex and edge sets.
ii. Generate a vertex degree set, which contains degree of all vertices.
iii. Updating edge set.

a. Find vertex with maximum degree.
b. Add vertex with maximum degree in edge cover set.
c. Delete vertex with maximum degree from vertex set.
d. Update edge set by removing edges connected to maximum degree vertex.
f. Update vertex degree set from updated edge set.
g. Repeat Step a to e till edge set not empty.

Step 3, Assign a single color to all vertices of independent set.

Step 4, Add independent set in to set of independent sets.

Repeat Step 2 to 4 till all vertices added into some independent sets.

There are three main steps of algorithm, which takes maximum time of execution.

1. While calculating degree of all vertices. (Step 2, ii)
2. When algorithm find maximum degree vertex. (Step 2, iii, a)
3. When algorithm updates edge set and prepare edge set for next iteration. (Step 2, iii, d)

Among all three steps, first steps executed each time whenever new independent set finding starts. If graph size is large than this step takes most of the time of algorithm execution. ECGCA calculate degree of vertices sequentially, one vertices at a time, as shown in Figure 1. To optimize degree calculation time, this paper incorporate some modification in ECGCA and proposed an new improved edge cover based graph coloring algorithm with parallel computing.

3. Proposed Algorithm

Parallel computing can be achieved by multiprocessor system, multi core processor[3] and through multithreading for single processor at software level. This paper introduced algorithm with parallel computing at software level. To optimize execution time of degree calculation step, multi threading concepts are used by proposed algorithm. Through multi threading proposed algorithm achieve concurrency in degree calculation. A single thread is assign to each vertex for calculation of degree. Each thread is initiated one by one and thread executed parallel, as shown in Figure 2.

Although for single processor system practically it is not possible to achieve exact parallelism. But using multithreading concept concurrency can be achieved. With concurrency algorithm can increase the CPU utilization and execute multiple thread by CPU scheduling algorithm of operating system.

4. Experimental Results

Proposed algorithm is implemented using Java programming language. Existing Sequential Algorithm (ECGCA) is also implemented using Java to compare the results of sequential and parallel algorithm. Jdk1.8.0_74 is used as java development kit and Windows 7 Ultimate 64 bit operating system is used to as a development and testing platform. Intel Core2 Duo CPU T657 @2.10 GHz with 2 GB memory computer system is used for performing experiments.

Graph instances provided by DIMACS (Discrete Mathematics and Theoretical Computer Science) are used as experimental data. DIMACS graph instances are generated by some practical application and some of them are generated randomly through computer programs. Graph data is available in the form of .col extension text files. These files contain various graph information like number of vertices and edges in graph. Edges information is given in the form of edge list. All the major applications and categories of graph instances are taken as experimental data like Mycile, Queen, Insertion, Full Insertion DSJC, DSJR, Wap, School, zeroin, GEOM, R100, and Holes problem graphs.

Table 1 shows the experimental results of various DIMACS graph instances. Experimental results contain manly two information numbers of colors generated by

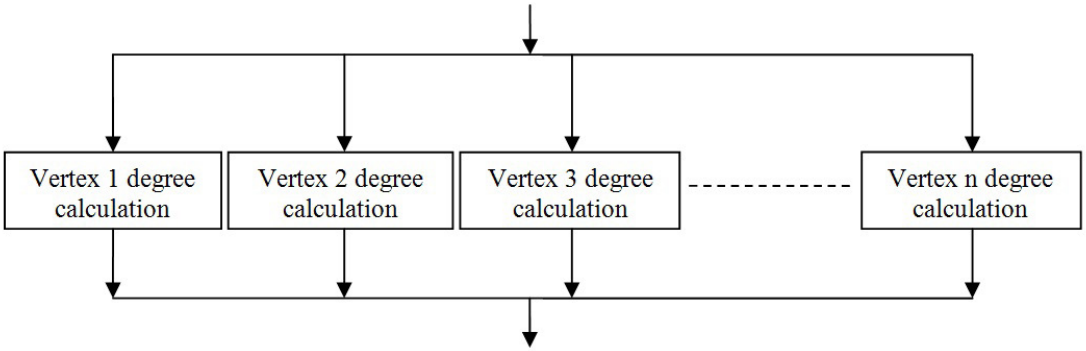

Figure 2. Degree calculation step of improved proposed algorithm.

Table 1. Experimental Results of Proposed Algorithm (Parallel ECGCA)

Instance	Vertices	Edges	Avg. Degree	Colors	Time(s)
myciel3	11	20	3.64	4	0.031
myciel4	23	71	6.17	5	0.063
queen5_5	25	320	25.60	7	0.076
1-FullIns_3	30	100	6.67	4	0.025
queen6_6	36	580	32.22	8	0.118
2-Insertions_3	37	72	3.89	4	0.037
myciel5	47	236	10.04	6	0.112
queen7_7	49	952	38.86	10	0.127
2-FullIns_3	52	201	7.73	5	0.063
3-Insertions_3	56	110	3.93	4	0.063
queen8_8	64	1456	45.50	12	0.203
1-Insertions_4	67	232	6.93	5	0.078
huck	74	602	16.27	11	0.140
4-Insertions_3	79	156	3.95	4	0.062
jean	80	508	12.70	10	0.124
3-FullIns_3	80	346	8.65	6	0.051
queen9_9	81	2112	52.15	14	0.312
1-FullIns_4	93	593	12.75	5	0.117
queen8_12	96	2736	57.00	16	0.485
queen10_10	100	2940	58.80	15	0.356
queen11_11	121	3960	65.45	18	0.702
DSJC125.9	125	6961	111.38	56	2.852
miles1500	128	10396	162.44	80	6.110
2-Insertions_4	149	541	7.26	5	0.123
2-FullIns_4	212	1621	15.29	6	0.395
DSJC250.9	250	27897	223.18	93	51.038
DSJC250.5	250	15668	125.34	41	9.622
3-Insertions_4	281	1046	7.44	5	0.308
1-FullIns_5	282	3247	23.03	6	0.709
3-FullIns_4	405	3524	17.40	8	0.677
DSJC500.9	500	224874	899.50	170	1171.148
DSJR500.1c	500	121275	485.10	103	541.442
DSJR500.5	500	58862	235.45	207	453.514
DSJC500.5	500	62624	250.50	71	184.888
DSJC500.1	500	12458	49.83	15	1.922
2-Insertions_5	597	3936	13.19	7	1.048
1-Insertions_6	607	6337	20.88	7	1.255
2-FullIns_5	852	12201	28.64	8	2.917
qg.order30	900	26100	58.00	40	42.266
wap05a	905	43081	95.21	58	89.764
wap06a	947	43571	92.02	58	89.254

DSJC1000.1	1000	49629	99.26	30	74.581
3-Insertions_5	1406	9695	13.79	7	3.033
abb313GPIA	1557	65390	83.99	15	57.373
3-FullIns_5	2030	33751	33.25	9	16.532
4-FullIns_5	4146	77305	37.29	10	70.042
qwhdec. order30. holes316.1	900	26100	58.00	38	42.783
qwhdec. order33. holes381.bal.1	1089	34848	64.00	45	78.003
qwhdec. order35. holes405.1	1225	41650	68.00	46	109.572
qwhopt. order18. holes120.1	324	5508	34.00	24	2.362
qwhopt. order30. holes320.1	900	26100	58.00	41	39.297
qwhopt. order33. holes381.bal.1	1089	34848	64.00	44	70.828
qwhopt. order35. holes405.1	1225	41650	68.00	50	102.389
qwhopt. order40. holes528.1	1600	62400	78.00	53	271.714
R100_1g	100	509	10.18	7	0.172
R100_1gb	100	509	10.18	7	0.124
R100_5g	100	2456	49.12	20	0.437
R100_5gb	100	2456	49.12	21	0.500
R100_9g	100	4438	88.76	45	1.264
R100_9gb	100	4438	88.76	46	1.249
r250.5	250	14849	118.79	102	19.012
R75_9gb	75	2513	67.01	37	0.631
school1	385	19095	99.19	43	14.510
school1_nsh	352	14612	83.02	39	8.332
wap01a	2368	110871	93.64	59	612.583
wap02a	2464	111742	90.70	60	630.470
wap05a	905	43081	95.21	57	81.830
wap06a	947	43571	92.02	57	83.672
wap07a	1809	103368	114.28	65	501.037
wap08a	1870	104176	111.42	65	510.938
will199GPIA	701	7065	20.16	11	1.918
zeroin.i.1	211	4100	38.86	50	0.914

zeroin.i.2	211	3541	33.56	31	0.516
zeroin.i.3	206	3540	34.37	30	0.535
GEOM90	90	531	11.80	9	0.171
GEOM90a	90	879	19.53	15	0.249
GEOM90b	90	950	21.11	17	0.234

Table 2. Chromatic number and execution time of sequential and parallel ECGCA

Instance	Vertices	Edges	Parallel ECGCA		Sequential ECGCA	
			Colors (K)	Time (s)	Colors (K)	Time (s)
DSJC1000.1	1000	49629	30	74.581	30	91.518
wap05a	905	43081	58	89.764	58	101.861
qwhopt.order33.holes381.bal.1	1089	34848	44	70.828	49	97.010
qwhdec.order35.holes405.1	1225	41650	46	109.572	49	141.216
wap06a	947	43571	57	83.672	57	101.860
3-Insertions_5	1406	9695	7	3.033	7	3.679
abb313GPIA	1557	65390	15	57.373	15	75.792
qwhopt.order40.holes528.1	1600	62400	53	271.714	60	357.825
wap07a	1809	103368	65	501.037	65	649.686
wap08a	1870	104176	65	510.938	68	651.112
3-FullIns_5	2030	33751	9	16.532	9	21.885
wap01a	2368	110871	59	612.583	59	807.874
wap02a	2464	111742	60	630.470	59	832.275
4-FullIns_5	4146	77305	10	70.042	10	104.195

algorithm, also known as chromatic number and execution time in seconds. Table 1 also contains graph instance name, number of vertices and number of edges in graph. This information is directly taken from the text file (.col file). Degree of graph is also calculated by number of vertices and number of edges in graph, average degree is also presented in Table 1.

5. Result Analysis

To know the improvement in algorithm, sequential version of algorithm is also implemented and experimented on the same platform for some graph instances.

In Table 2, execution time and chromatic numbers are compared of both sequential and parallel edge cover based graph coloring algorithms for some large graphs. By analyzing the results in Figure 3, it is clearly observed that parallel version of algorithm executed time is optimum for large graphs. In table 2 total 14 sample graphs are considered to analyze execution time.

Average execution time of sequential version of algorithm T(Sequential)= 336.482 Seconds (1)

Table 3. Chromatic number (Number of Colors) generated by Parallel and Sequential ECGCA

Instance	Vertices	Edges	Parallel ECGCA	Sequential ECGCA
			Colors (K)	Colors (K)
queen6_6	36	580	8	10
queen7_7	49	952	10	12
queen8_8	64	1456	12	14
R75_9gb	75	2513	37	39
GEOM90	90	531	9	10
GEOM90a	90	879	15	16
GEOM90b	90	950	17	18
queen8_12	96	2736	16	15
queen10_10	100	2940	15	17
R100_1g	100	509	7	8
R100_1gb	100	509	7	8
R100_5g	100	2456	20	22
R100_5gb	100	2456	21	22

Average execution time of parallel version of algorithm T(Parallel)= 258.511 Seconds (2)

By the (1) and (2) it is clearly observed that execution time is reduced by 23.17 % in parallel version for large graphs.

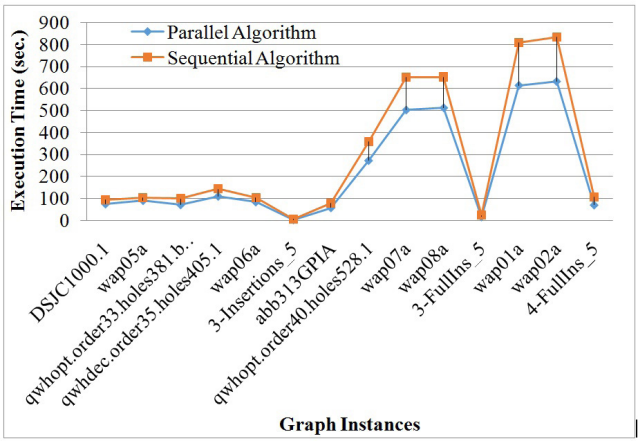

Figure 3. Execution time of Sequential and Parallel ECGCA.

Table 3 shows the chromatic number comparison of sequential and parallel ECGCA for small and medium size graphs. By the Figure 4, it has been observed that proposed parallel ECGCA generates better chromatic number.

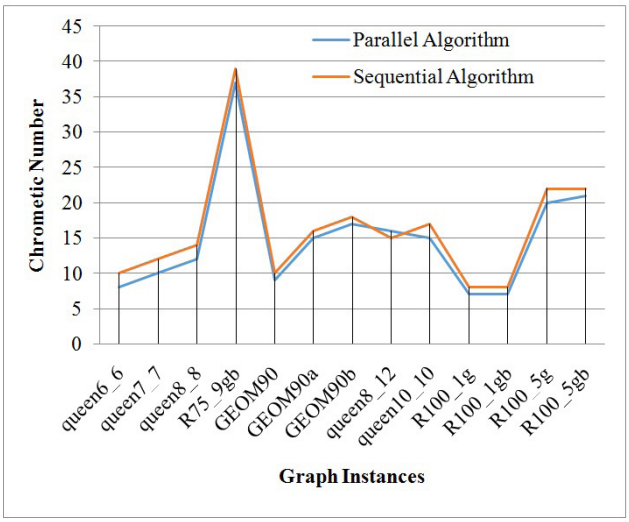

Figure 4. Chromatic number comparison of Sequential and Parallel ECGCA.

While experimenting with sequential ECGCA it has been found that number of colors (Chromatic number) is fixed in different attempts of algorithm and even there no changes in independent sets also for example, when graph instance queen5_5.col executed through sequential ECGCA result generated is shown in Table 4. If same graph instance execution is attempt again, result is same as shown in table 4. There are no changes even in number of colors or in independent sets.

Table 4. Sequential ECGCA results of queen5_5 graph instance in different attempt of execution

Set Number	Number of Vertices in Set	Vertices
1	4	3,6,20,22
2	5	4,7,15,18,21
3	4	2,10,16,24
4	4	5,8,11,19
5	4	1,9,12,23
6	3	14,17,25
7	1	13

While experimenting with parallel ECGCA one interesting fact is observed, that parallel algorithm gives different chromatic number for the same graph instance in different attempts and independent set containing vertices can be very in different attempts. For example graph instance queen5_5.col generated the result as shown in Table 5 in first attempt. Same graph instance result is different in another attempt, as shown in Table 6.

Similarly, this paper shows the 10 graph instances results in different attempts (15 attempts per graph) in Table 7.

Table 5. Parallel ECGCA results of queen5_5 graph instance in first attempt of execution

Set Number	Number of Vertices in Set	Vertices
1	4	2,10,16,24
2	5	3,6,14,17,25
3	5	4,7,15,18,21
4	5	1,9,12,20,23
5	5	5,8,11,19,22
6	1	13

Table 6. Parallel ECGCA results of queen5_5 graph instance in another attempt of execution

Set Number	Number of Vertices in Set	Vertices
1	4	7,15,16,24
2	4	3,6,20,22
3	5	2,9,11,18,25

4	4	1,10,12,23
5	3	8,19,21
6	2	4,17
7	2	5,14
8	1	13

6. Discussion

Sequential and parallel both algorithms are implemented using Java programming language for experiment. Some common features like file handling and collection framework classes are used in both the implementations. But to achieve parallelism in parallel ECGCA multithreading concept of java is used in implementation. Multithreading overhead also affected the time complexity of algorithm. So in experiments it is observed that when both algorithms are tested on small size graphs as results shown in Table 8. It is found that parallel algorithm takes more time to execute than sequential algorithm. But in some cases for small size graphs it has been observed that parallel algorithm give better chromatic number as shown in

Table 7. Parallel ECGCA results of 15 attempts per graph on 10 graph instances

Instance	Total attempts	Variation 1		Variation 2		Variation 3		Variation 4	
		Attempts	K	Attempts	K	Attempts	K	Attempts	K
queen5_5	15	2	7	11	8	2	9		
queen6_6	15	1	8	5	9	6	10	3	11
queen7_7	15	3	10	6	11	5	12	1	13
queen8_8	15	1	12	9	13	5	14		
R75_5g	15	1	15	4	16	8	17	2	18
R75_9gb	15	2	36	4	38	4	39	5	40
queen9_9	15	5	14	8	15	2	16		
GEOM90	15	6	9	9	10				
GEOM90a	15	3	15	10	16	2	17		
R100_5g	15	8	20	7	21				

Table 8. Execution time and chromatic number of parallel vs Sequential ECGCA for small size graphs

Instance	Vertices	Edges	Parallel ECGCA		Sequential ECGCA	
			Colors (K)	Time (s)	Colors (K)	Time (s)
myciel3	11	20	4	0.031	4	0.010
myciel4	23	71	5	0.063	5	0.018
queen5_5	25	320	7	0.076	7	0.061
1-FullIns_3	30	100	4	0.025	4	0.031
queen6_6	36	580	8	0.118	10	0.087
2-Insertions_3	37	72	4	0.037	4	0.029
Myciel5	47	236	6	0.112	6	0.057
queen7_7	49	952	10	0.127	12	0.110
2-FullIns_3	52	201	5	0.063	5	0.030
3-Insertions_3	56	110	4	0.063	4	0.036
queen8_8	64	1456	12	0.189	14	0.133
1-Insertions_4	67	232	5	0.078	5	0.029
R75_1g	70	251	6	0.116	6	0.050
huck	74	602	11	0.146	11	0.107
R75_5g	75	1407	16	0.282	16	0.159

table 8, like in case of graph instances *queen6_6, queen7_7* and *queen8_8*.

7. Conclusion and Future Scope

Capability of parallel ECGCA to solve the graph coloring problem for large graphs will provide great incentive in the field of solving graph coloring problem. Proposed improvement in algorithm produce results in optimum time. Capability of solving problem in optimum time will help the real time application, which requires result in minimum delay like social networking sites. Proposed algorithm is tested and evaluated on different benchmark data sets of DIMACS. Most of the graph coloring application data sets are successfully tested. Most of the time high execution success rate is achieved in experiments. So the parallel ECGCA is efficient, optimum time complex, large dataset supported and wide area of application supported algorithm.

As shown in result analysis section (Section 5) in this paper, behavior of algorithm in different attempts of execution is discussed. It has been observed that algorithm generate different results for same graph instance in different attempts. In some attempts algorithm giving better chromatic number, so in future this algorithm can be analyze in different attempts and some facts can be extracted for better chromatic number. This fact can be used to select vertices sequence of processing to get better results.

Proposed algorithm is also tested on small size graphs, but due to multithreading overhead execution time increased in parallel ECGCA. So in future some better option can be used, so that algorithm can give better results for small size graphs also.

8. References

1. Gandhi NM, Misra R. Performance comparison of parallel graph coloring algorithms on BSP model using hadoop. Proceeding of International Conference on Computing Networking and Communications (ICNC); USA; 2015. p. 110–6. Crossref
2. Duan F, Lei Y, Yu L, Kachker RN and Kuhn DR. Improving IPOG's vertical growth based on a graph coloring scheme. Proceeding of 8th International Conference on Software Testing Verification and Validation Workshops (ICSTW); Austria; 2015. p. 1–8. Crossref
3. Stecke K. Design planning, scheduling and control problems of flexible manufacturing systems. Annals of Operations Research. 1985 Jan; 3(1):1–12. Crossref
4. Camino JT, Mourgues S, Artigues C, Houssin L. A greedy approach combined with graph coloring for non-uniform beam layouts under antenna constraints in multibeam satellite systems. Proceeding of 7th Advanced Satellite Multimedia Systems Conference (ASMS) and 13th Signal Processing for Space Communications Workshop (SPSC); Italy; 2014. p. 374–81. Crossref
5. Barnier N, Brisset P. Graph coloring for air traffic flow management. Annals of Operations Research. 2004 Aug; 130(1):163–78. Crossref
6. Robson JM. Algorithms for maximum independent sets. Journal of Algorithms. 1986 Sep; 7(3):425–40. Crossref
7. Luby M. A simple parallel algorithm for the maximal independent set problem. SIAM Journal on Computing. 1986 Nov; 15(4):1036–53. Crossref
8. Krivelevich M, Nathaniel R, Sudakov B. Approximating coloring and maximum independent sets in 3-uniform hypergraphs. Journal of Algorithms. 2001 Oct; 41(1):99–113. Crossref
9. The Independent Set Algorithm [Internet]. 2006 [updated 2006; cited 2017 Mar 3]. Available from: http://www.dharwadker.org/independent_set
10. Keil J. Mark, Mitchell J, Pradhan D, Vatshell M. An algorithm for the maximum weight independent set problem on outerstring graphs. Proceeding of 27th Conference on Computational Geometry; Canada. 2016 May; 60:19–25.
11. A family of greedy algorithms for finding maximum independent set [Internet]. 2015 [updated 2015 May 4; cited 2017 Mar 17]. Available from: https://arxiv.org/abs/1505.00752
12. Accelerating local search for the maximum independent set problem [Internet]. 2016 [updated 2016 Feb 4; cited 2017 Mar 17]. Available from: https://arxiv.org/abs/1602.01659