Maximum Clique Conformance Measure for Graph Coloring Algorithms

Abdulmutaleb Alzubi
Jadarah University
Dept. of Computer Science
Irbid, Jordan

Mohammad Al-Haj Hassan
Zarqa University
Dept. of Computer Science
Zarqa, Jordan

Mohammad Malkawi
Jordan University for Science and Technology,
Dept. of Software Engineering, Irbid, Jordan

ABSTRACT
The Graph Coloring Problem (GCP) is an essential problem in graph theory, and it has many applications such as the exam scheduling problem, register allocation problem, timetabling, and frequency assignment. The maximum clique problem is also another important problem in graph theory and it arises in many real life applications like managing the social networks. These two problems have received a lot of attention by the scientists, not only for their importance in the real life applications, but also for their theoretical aspect. Solving these problems however remains a challenge, and the proposed algorithms for this purpose apply to rather small graphs, whereas many real life application graphs encompass hundreds or thousands of nodes. This paper presents a new measure for evaluating the efficiency of graph coloring algorithms. The new measure computes the clique conformance index (CCI) of a graph coloring algorithm. CCI measures the rate of deviation of a coloring algorithm from the maximum clique during the process of coloring a graph. The paper presents empirical measurement for two coloring algorithms proposed by the authors.

General Terms
Algorithms, Graph Coloring

Keywords
Algorithms, Clique conformance Index, Convergence Rate, Deviation Rate, Graph Coloring, Maximum Clique.

1. INTRODUCTION
An undirected graph G is an ordered pair (N, E) where N is a set of Nodes and E is a set of non-directed Edges between nodes. Two nodes are said to be adjacent if there is an edge between them. An edge can carry a weight in weighted graphs; for example in a graph used for exam scheduling, the weight of an edge can be used to specify the number of students registered in two adjacent classes. The importance of studying graphs comes from their ability to model many real life problems, such as computer and telecomm networks, social networks, resource allocation problems, and many more [1, 4, 12].

Graph coloring is the process of labeling the graph's nodes (or edges) with special labels (colors) such that no two adjacent nodes (or edges) have the same color. An interesting coloring problem is the four colors theorem, which colors certain planar graphs with exactly four colors. Another interesting problem is the problem of Clique coloring, which is a variation of the classical vertex coloring graph coloring. Clique coloring requires that only every inclusion-wise maximal (not extendable) clique contains at least two different colors [2], instead of requiring that the two end points of each edge have two different colors.

Clique in an undirected graph G is a subset of nodes N such that for every two nodes in N, there exists an edge connecting both nodes. The maximum clique is that one which has the maximum number of nodes. Finding the minimum number of colors for a given graph or finding the maximum clique in a graph is considered NP-complete, i.e., they run in a non-polynomial time. Detecting the maximum clique of a graph can be very useful for finding a minimal coloring for a graph, where a *k*-clique-colorable graph has a *k* chromatic number [2]. Although, this may not hold for all graphs, e.g., 2-clique and 3-clique-colorable graphs, the k-clique-colorable can be a reasonable bound for the chromatic number of a graph [2]. Hence, if a coloring heuristic algorithm manages to first color the nodes in a clique with *k* colors (*k*-clique-colorable) before moving to other nodes, then the algorithm is more likely to find a minimal coloring of the graph.

The problem of finding the maximum clique in an undirected graph is considered one of the NP-complete problems. This means that there is no known algorithm for solving this problem in a polynomial time, except for those which have been developed for specialized graphs such as the planer graphs or perfect graphs where the problem can be solved in a polynomial time [2, 6, 7, 8, 14, 15]. Solving this problem can also be done in polynomial time if k is constant, where k is the number of nodes in the clique; in this case all subgraphs of at least k nodes will be checked whether they form a clique or not. This method is called the brute force algorithm.

The main objective of this paper is to develop a measure to evaluate the level of proximity of coloring algorithms to the maximum clique detection algorithm. The measure will show if the coloring algorithm colors the nodes of the maximum clique first, and if not, it will show the level of deviation from the clique. This measure will be used to evaluate the efficiency of two coloring algorithms. The coloring algorithms have been tested against special graphs such as the 4-colorable graphs, triangle free girth graphs and others [12]. Test graphs are generated randomly, using a tool developed by the researchers to generate random graphs with varying densities.

Section 2 will describe the *clique conformance index* measure (CCI). Section 3 will describe the two coloring algorithms used in this study for evaluation under the CCI measure. Section 4 will present the experiment environment and results. Conclusions will be presented in section 5.

2. CLIQUE CONFORMANCE INDEX
In order to evaluate the efficiency of graph coloring algorithms a new measure is proposed, which measures the ability of the coloring algorithm to assign colors to the nodes

of a clique before it moves to other nodes in the graph, starting with the largest clique first. This measure is called the *clique conformance index* (CCI). The purpose of this new measure is not to detect the cliques in each subgraph; rather its purpose is to analyze the efficiency of the coloring algorithms in terms of how closely each algorithm follows the maximum clique-based coloring of a graph. The definition and computation of CCI is given below.

The clique of each subgraph will be fully recognized and detected using a brute force algorithm. Then, the coloring path taken by the coloring algorithm is traced and tested against the cliques of the graph. A complete matching exists if the algorithm colors all the nodes of the clique before it moves to another subgraph. In this case the algorithm will consume *k* colors for a *k*-colorable clique. A partial matching exists when the algorithm colors nodes from the clique and then moves to another set of nodes before it returns back to color the rest of the clique nodes. In this case, the algorithm is not guaranteed to use only *k* colors. The efficiency of the algorithm will be determined by the rate of convergence between the set of colored nodes and the nodes of the clique and by the distance between the nodes colored outside the clique and the nodes of the clique.

The rate of convergence (ρ) is defined as follows. Let the number of nodes in a clique be *k* and let the coloring algorithm colors *l* nodes, where $l \leq k$ and all *l* nodes are included in the set of *k* nodes. The algorithm will use *l* colors to color the *l* nodes. Then the algorithm begins to color nodes outside the clique. The rate of convergence is given by $\rho = l/k$, which measures the ability of the algorithm to color clique nodes first.

In order to account for the number of extra colors consumed due to moving away from the clique nodes, the deviation rate of each node colored outside the clique to which the node belongs is computed. The deviation rate of node N_i in graph G is defined as follows. Let N_i be part of the clique nodes, i.e., ($N_i \in k$). Define the order of coloring a node $R(N_i)$ to be the sequence order of coloring node N_i; so if node N_i is assigned a color after coloring R-1 nodes, then the order of coloring N_i is $R(N_i)$. The deviation rate of node N_i is given by $\delta = (R(N_i)-k)/R(N_i)$, where k is the size of the clique and $R(N_i)$ is the order of coloring node (N_i). δ is set to zero if $R(N_i) < k$.

For example, assume that the size of a clique is 4 in a 7 nodes graph (k=4); the clique nodes are {1, 2, 3, 4}. Assume that the coloring algorithm colors the graph in the following sequence (1, 2, 5, 6, 3, 7, 4). Two clique nodes (1 and 2) are assigned colors before the algorithm skips to node 5 which does not belong to the clique. Hence, the convergence rate $\rho = 2/4 = 0.5$. The deviation rate for nodes 1 and 2 is zero since the order of these nodes is smaller than the clique size. And the deviation rate for node 3 (colored outside the clique sequence) is given by $\delta = (5-4)/6 = 0.17$. The deviation rate of node 4 is $\delta = (7-4)/7 = 0.43$. The average deviation rate for the clique nodes is:

$$\Delta = \sum_{i=1}^{k} \delta i / k = (0 + 0 + 0.17 + 0.43)/4 = 0.15 \qquad (1)$$

If the order of coloring for nodes {1, 2, 3, 4} was less than or equal to four, then the convergence rate ρ will be 4/4 = 1 and the average deviation rate Δ will be zero.

The *Clique Conformance Index* (CCI) combines both the convergence and deviation rates into one index. CCI measures the overall efficiency of the algorithm and is defined as follows: $CCI = \rho/\Delta$. CCI measures the ability of the algorithm to stick to the clique nodes and how quickly it returns to the clique if it wanders away from it. Note that CCI increases when more nodes are colored from within the clique nodes (larger values of ρ) and when the algorithm does not move far before it returns to the clique (smaller values of Δ). In the above example, CCI = 0.5/0.15 = 3.6. The larger the value of CCI, the better is the algorithm. CCI is set to N when Δ=0. CCI easily accounts for odd cases, such as when the convergence rate is very high say 0.9 and the deviation rate of this node is very large, say 0.9 (CCI is 0.9/0.9 =1). This gives a very low conformance index. However, if the node distance is small, say 0.1, then CCI is 0.9/0.1 = 9, which means that the algorithm had returned to the clique quickly and now it can resume the coloring of the nodes in the clique.

Next, the CCI measure is applied to two algorithms, namely the largest degree algorithm and the largest degree with saturation algorithm.

3. EVALUATION of TWO GRAPH COLORING ALGORITHMS

In [12], the authors introduced what they called "largest Degree Coloring (LDC) Algorithm for Exam Scheduling using Graph Coloring". That algorithm depends on an approach that employs the notion of the Largest Degree to perform the required task. It is given here for completeness, and For the purpose of comparison between it and the new algorithm prosed in this research:

3.1 Largest Degree Coloring (LDC) Algorithm

1. Sort nodes based on degree in descending order.
2. Select the first node in the list; Color the node with smallest available color
3. List the neighbors of the selected node
4. Sort the neighbors of the selected node in descending order based on degree, if two or more nodes have the same degree, then the one with the largest ID is ordered first.
5. Color the neighbors of the selected node, starting with the first node in the list, for each node; check all its neighbors which have already been colored. Color the node with the smallest available color
6. When all neighbors of the selected node have been colored; go to the next node in the main list of nodes
7. Go to step 3

Stop when all nodes have been colored.

The LDC has been shown to succeed in coloring some special graphs with the minimum number of colors [12]. The algorithm colors one of the famous 5-coloring graphs with 4 colors. The algorithm also colors all 4-coloring planer graphs with exactly 4-colors. 6-triangulation graphs (known to be 5-colorables) will be colored by LDC with exactly 5 colors. Triangle free girth graphs are colored by LDC with the smallest possible number of colors, i.e., the chromatic number.

The performance of LDC for randomly generated large graphs is yet to be investigated. One of the driving motives for proposing the algorithm was to derive exam schedules with minimal conflicts. The intuition was to detect a class whose students are registered in many other classes such that

conflicts arise when scheduling classes to the same time slot (or color). The LDC was shown to produce efficient exam schedules [12]. However, it is not possible to generalize this result for very large graphs with varying densities. Therefore, the performance of this algorithm will be tested using the CCI measure. It has been shown that the complexity of the LDC is $O(n^2)$ [12].

When two or more nodes have the same degree, the largest degree algorithm uses the node ID to resolve the conflict. This is a simple solution, but has no relevance to the strength or weakness of the algorithm. A more efficient criterion is to select the node with the maximum saturation degree [11]. The term saturation degree (SD) refers to the number of differently colored nodes adjacent to a particular node. For example, if node N_i has 5 neighbors of which two are colored with two different colors, the $SD(N_i)=2$. Now if $SD(N_j) = 3$, then N_j is said to have a larger saturation degree. Intuitively N_j should be colored first because it has fewer choices than N_i.

The LDC algorithm described above is modified and the maximum saturation degree criterion is used to choose between two nodes having the same degree. This algorithm is called the Largest Degree with Saturation Coloring (LDSC).

3.2 Largest Degree with Saturation Coloring (LDSC) Algorithm

1. Sort nodes based on degree in descending order.
2. Select the first node in the list; Color the node with smallest available color.
3. List the neighbors of the selected node.
4. Sort the neighbors of the selected node in descending order based on degree, if two nodes have the same degree, choose the node with the greater saturation degree.
5. Color the neighbors of the selected node, starting with the first node in the list. For each node; check all its neighbors which have already been colored. Color the node with the smallest available color
6. When all neighbors of the selected node have been colored; go to the next node in the main list of nodes
7. Go to step 3
8. Stop when all nodes have been colored.

In terms of complexity, LDSC requires one more iteration per node in order to find the saturation degree, thus increasing the complexity of LDSC to $O(n^3)$.

In the next section, the experiments used for the evaluation of the coloring algorithms using the CCI measure with heavy, medium, and low densities respectively are described.

4. PERFORMANCE EVALUATION

A JAVA program has been developed to generate random graphs with varying sizes and densities. The size of a graph, i.e., the number of nodes in a graph is selected by the user. Three densities are provided for each graph (heavy, medium, and low). The density of the graph is determined by the probability of two nodes being connected by a link. The probabilities used in this program are 0.75, 0.5, and 0.25 for

The JAVA program implements the LDC and LDSC coloring algorithms besides the brute force method used for the detection and enumeration of the nodes of maximal clique in the graphs. The program will compute the convergence rate, the deviation rates for each node of the clique, and the clique conformance index CCI for each run.

The results of the test are shown in Table 1a and Table 1b. The results in Table 1a and Table 1b are the average of several runs for LDC and LDSC, where:

1. The number of nodes (input by the user).
2. Size of the largest clique: generated by the brute force method.
3. Number of colors: the number of colors used to color the graph.
4. ρ, Δ and CCI: computed by the program

Both tables has three sections for low, medium and high densities.

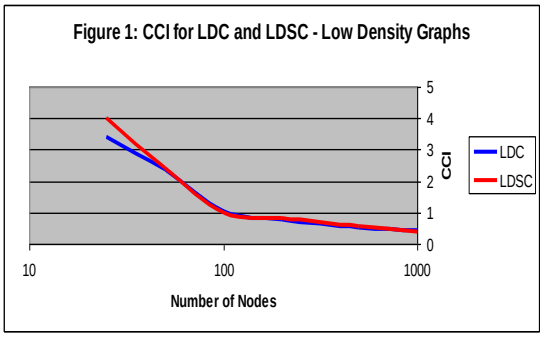
Table 1a: Summary of results of Algorithm LDC

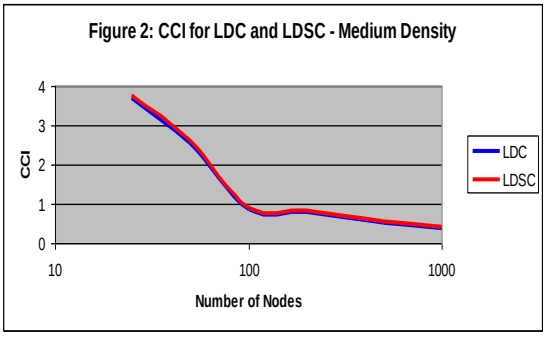
Low Density P=0.25		**LDC Algorithm**			
Number of Nodes	Size of the clique	Coloring	Converg-ence Rate □	Average Deviation Δ	CCI = □/Δ
25	4	5	0.92	0.11	3.41
50	4	7	0.75	0.30	2.37
100	4	12	0.68	0.67	1.02
200	5	20	0.58	0.73	0.78
500	5	39	0.59	0.76	0.53
1000	6	66	0.31	0.72	0.44
Mediumr Density P=0.5		**LDC Algorithm**			
Number of Nodes	Size of the clique	Coloring	Converge nce Rate □	Average Deviation Δ	CCI = □/Δ
25	5	7.00	0.87	0.26	3.69
50	5	11.67	0.81	0.41	2.54
100	7	21.33	0.49	0.57	0.86
200	8	36.00	0.61	0.77	0.78
500	9	73.33	0.44	0.80	0.51
1000	11	129.00	0.28	0.72	0.38
High Density P=0.75		**LDC**			
Number of Nodes	Size of the clique	Coloring	Converge nce Rate □	Average Deviation Δ	CCI = □/Δ
25	9	11	0.92	0.14	6.29
50	12	19	0.67	0.44	1.53
100	14	32	0.71	0.51	1.39
200	16	56	0.70	0.59	1.06
500	20	123	0.39	0.66	0.72
1000	21	218	0.38	0.67	0.56

Figures 1, 2 and 3 show the clique conformance index for LDC and LDSC for low, medium and high density graphs respectively. For lower number of nodes, both algorithms conform more closely to the maximum clique in the graphs. from the maximum clique. The LDSC performs slightly better than the LDC algorithm for low and medium density graphs. The performance improvement of LDSC is more visible for heavy density graphs. Although, both algorithms perform poorly when the number of nodes exceeds 100 nodes.

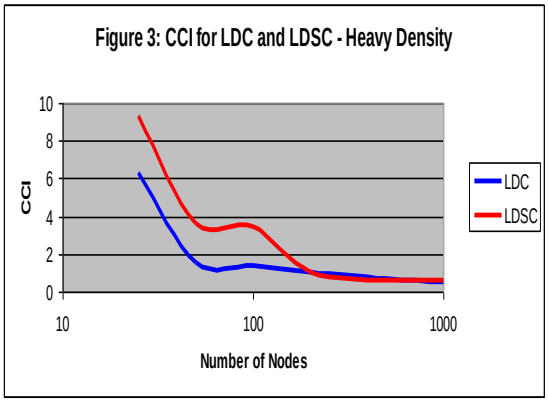
Table 1b: Summary of results of Algorithm LDSC

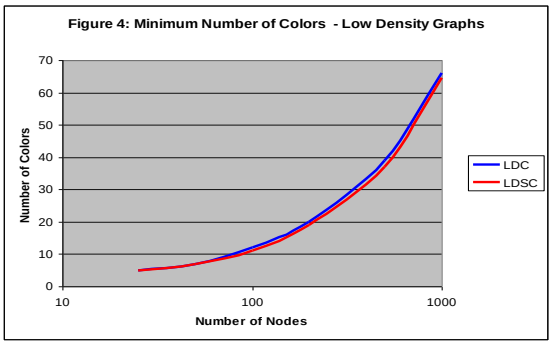
Low Density P=0.25		**LDSC Algorithm**			
Number of Nodes	Size of the clique	Coloring	Converg-ence Rate □	Average Deviation Δ	CCI = □/Δ
25	4	5.00	1.00	0.00	4.00
50	4	7.00	0.75	0.28	2.41
100	4	11.00	0.68	0.67	1.01
200	5	19.00	0.58	0.71	0.80
500	5	37.00	0.40	0.74	0.54
1000	6	64.67	0.31	0.74	0.40
Mediumr Density P=0.5		**LDSC Algorithm**			
Number of Nodes	Size of the clique	Coloring	Converge nce Rate □	Average Deviation Δ	CCI = □/Δ
25	5	4.79	0.93	0.21	3.75
50	5	9.70	0.81	0.38	2.63
100	7	15.10	0.49	0.55	0.90
200	8	34.33	0.61	0.73	0.82
500	9	39.42	0.44	0.79	0.55
1000	11	126.00	0.28	0.72	0.41
High Density P=0.75		**LDSC Algorithm**			
Number of Nodes	Size of the clique	Coloring	Converge nce Rate □	Average Deviation Δ	CCI = □/Δ
25	9	11	1.00	0.00	9.33
50	12	18	0.70	0.41	3.62
100	14	31	0.76	0.57	3.41
200	16	55	0.65	0.61	1.06
500	20	118	0.38	0.61	0.59
1000	21	214	0.40	0.69	0.57

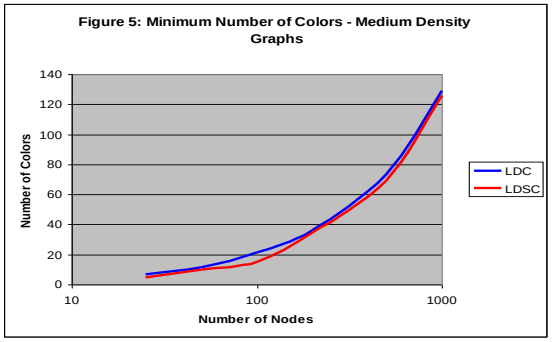




Similar behavior is observed for the number of colors consumed by LDC and LDSC algorithms. The results are demonstrated in Figures 4, 5 and 6. In general, LDSC consumes less number of colors than LDC. This is consistent with the observation that LDSC has a better CCI.







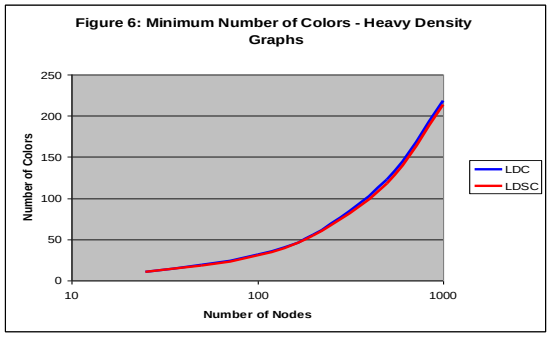


Figure 7 shows the number of colors consumed by LDSC and LDC for 1000 nodes and for low, medium and heavy density graphs.

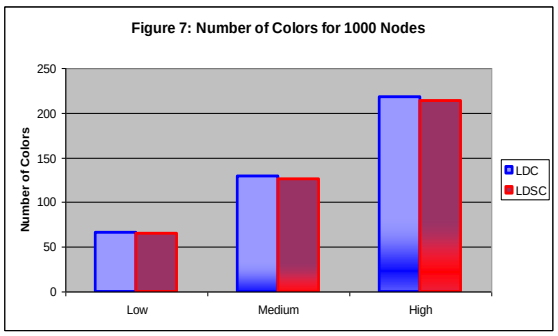


Table 2 ((a), (b), (c)) shows the chromatic number for small graphs (less than 50 nodes) computed using exhaustive search method, and the coloring provided by LDC and LDSC. Both LDC and LDSC achieved the minimal coloring as the chromatic number provided by the exhaustive search method. The CCI index for both LDC and LDSC for these small graphs is relatively large (3, 4 and 6 for low, medium and heavy density graphs). This result supports the premise that a large CCI index is likely to produce minimal coloring schema by the algorithms.

(a)		LDC	
Number of Nodes	Density	Number of Colors	Time in Seconds
15	L	4	Less than 1
16	L	4	Less than 1
17	L	3	Less than 1
18	L	3	Less than 1
19	L	4	Less than 1
20	L	4	Less than 1
25	L	4	Less than 1
30	L	4	Less than 1
40	L	5	Less than 1

(b)		LDSC	
Number of Nodes	Density	Number of Colors	Time in Seconds
15	L	4	Less than 1
16	L	4	Less than 1
17	L	3	Less than 1
18	L	3	Less than 1
19	L	4	Less than 1
20	L	4	Less than 1
25	L	4	Less than 1
30	L	4	Less than 1
40	L	5	Less than 1

(c)		BackTracking	
Number of Nodes	Density	Number of Colors	Time in Seconds
15	L	4	2 Seconds
16	L	4	4 Seconds
17	L	3	26 Seconds
18	L	3	1.14 minutes
19	L	4	11.38 minutes
20	L	4	40 minutes
25	L	4	1 Hour
30	L	4	1.3 Hour
40	L	5	3 Hours

Table 2: Comparing Between (a) LDC Algorithm and (b) LDSC Algorithm with (c) Backtracking

Unfortunately, and in this current research, the coloring schema for larger graphs could not be compared since the time complexity for the exhaustive search method increases exponentially as the number of nodes increases. Notice that Table 2 shows the coloring time for LDC and LDSC less than 1 second for 40 nodes or smaller graphs.

5. CONCLUSION

This paper presented the clique conformance index (CCI) as a new measure for the efficiency of coloring algorithms. CCI is used to measure the efficiency of two coloring algorithms, the largest degree (LDC) and the largest degree with saturation (LDSC) algorithms. Both algorithms perform well for small graphs; they produce large CCI and use minimal number of colors. For larger graphs, both algorithms conform poorly to the maximum clique. The coloring numbers with the minimal (chromatic number) colors were not be compared due to time complexity associated with the exhaustive coloring method. However, the low CCI suggests that both LDC and LDSC use more colors than the minimal coloring.

Further investigation of CCI is required to establish more accurate correlation between CCI and the minimal coloring of a graph. This is very useful since the time complexity for finding the minimal number of colors for large graphs is prohibitive. Therefore, a promising future work is to develop more efficient version of LDSC algorithm and develop it to deal with large graphs.

6. REFERENCES

[1] Allen, M., Kumaran, G., & Liu, T. (2002). A combined algorithm for graph-coloring in register allocation, *Proceedings of the Computational Symposium on Graph Coloring and its Generalizations*, pp. 100–111.

[2] Bacs´o, G. S., et al , Coloring the maximal cliques of graphs. SIAM J. Discrete Math., Vol. 17, No. 3, pp. 361–376, 2004.

[3] Balas, E., & Yu, C. S. (1986). Finding a maximum clique in an arbitrary graph. *SIAM J. Comput* , Vol 15, No. 4, pp 1054-1068.

[4] Balasundaram, B ., and Butenko, S., (2011), Clique relaxations in social network analysis: The maximum k-plex problem, Operations Research, Vol. 59, Issue 1, pp. 133–142.

[5] Caramia, M., & Dell'Olmo, P. (2001). Iterative Coloring Extension of a Maximum Clique. *Naval Research Logistics (NRL)*, Vol. 48, Issue 6, pp. 518–550.

[6] D´efossez, D., (2006), Clique-coloring some classes of odd-hole-free graphs. Journal of Graph Theory, Vol. 53, Issue 3, pp. 233–249.

[7] Duffus, D., et al, (1991), Two-colouring all two-element maximal anti-chains. J. Combinatorial Theory Ser. A, Vol. 57, Issue 1, pp. 109–116.

[8] Gravier, S., Ho`ang, C. T., and Maffray, F., (2003), Coloring the hypergraph of maximal cliques of a graph with no long path, Discrete Math., Vol. 272, No. (2-3), pp. 285–290.

[9] Hosseini, S.H. , et al., (1990). Analysis of a Graph Coloring Based Distributed Load Balancing Algorithm. *Journal of Parallel & Distributed Systems* , Vol. 10. , Issue 2, pp. 160–166

[10] Jensen, T. R., & Toft, B. (1994). *Graph Coloring Problems*, John Wiley & Sons, USA.

[11] Magnus, M. H. (1991). Frugal methods for the independent set and graph coloring problems. *PhD thesis*,

The State University of New Jersey, New Brunswick, New Jersey.

[12] Malkawi, M., Hajhassan, M., & Hajhasan, O. (2008). New exam scheduling algorithm using graph coloring. *International Arab Journal for IT (IAJIT)Vol. 5, No. 1, pp 80-86*.

[13] Robson, J. M. (1986). Algorithms for maximum independent set . *J. Algorithms* , Vol. 7, pp 425-440.

[14] Tomita, A., Tanaka, A., & Takahashi, H. (2006). The worst-case time complexity for generating all maximal cliques and computational experiments, Theoretical Computer Science 363 (2006), pp 28 – 42.

[15] Xiao, M., (2010), A Simple and Fast Algorithm for Maximum Independent Set in 3-Degree Graphs, Proceedings of the 4th International Workshop, WALCOM: Algorithms and Computations, pp. 281-292.