Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 9 (2012) 321 – 327

Procedia Computer Science

International Conference on Computational Science, ICCS 2012

Comparative Performance of Modified Simulated Annealing with Simple Simulated Annealing for Graph Coloring Problem

Anindya Jyoti Pal[a*], Biman Ray[b], Nordin Zakaria[a], Samar Sen Sarma[c]

[a]*High Performance Computing Centre, Universiti Teknologi Petronas Perak, 31750, Malaysia*
[b]*Dept of Comp. Sc. & Engg. Heritage Institute of Technology, Kolkata, 700107, India*
[c]*Dept of Comp. Sc. & Engg. University College of Sc & Tech, University of Calcutta, Kolkata, 700009, India*

Abstract

The problems which are NP-complete in nature are always attracting the computer scientists to develop some heuristic algorithms, generating optimal solution in time-space efficient manner compared to the existing ones. Coloring of the vertices of a graph with minimum number of colours belongs to the same category, where the algorithm designers are trying to propose some new algorithms for better result. Here, we have designed modified Simulated Annealing (MSA) for optimal vertex coloring of a simple, symmetric and connected graph (GCP). The algorithm has been tested upon a series of benchmarks including large scale test case and has shown better output than the simple or non-modified version of the same algorithm. This paper describes the advancement of performance of simple SA applied upon the problem of graph coloring using a specially designed operator called random change operator instead of the general change operator. Our work is still going on for designing better algorithms generating optimal solutions.

Keywords: NP; SA; MSA; GCP

1. Introduction

A *linear graph* (or simply a *graph*) G = (V, E) consists of a set of objects V = {v_1, v_2, ...}called vertices in G, and another set E = {e_1, e_2,….}, whose elements are called edges in G, such that each edge e_k is identified with an unordered pair (v_i, v_j) of vertices. Proper coloring of a simple, symmetric and connected graph is a classical problem of graph theory. A k-coloring of G is a partition of V into k subsets C_i, $i = 1$ ..... k, such that no adjacent nodes belong to the same subset. The k–colorability optimization problem is to find a k-coloring of G with k as small as possible. This smallest k corresponds to the chromatic number $\chi(G)$ of graph G. A k-vertex coloring of any graph is an assignment of k–colors 1, 2, 3,……..k to the vertices of G.

The reasons why the graph coloring problem is important are twofold. First, there are several areas of practical interest, in which the ability to color an undirected graph with a small number of colors as possible, has direct influence on how efficiently a certain target problem can be solved. Timetable scheduling [1], examination scheduling [2], register allocation [3], printed circuit testing [4], electronic bandwidth allocation [5], microcode optimization [6], channel routing [7], the design and operation of flexible manufacturing systems [8], computation

* Corresponding author. Tel.: +6013517441;
E-mail address: anindyajp@gmail.com.

doi:10.1016/j.procs.2012.04.034

of sparse Jacobian elements by finite differencing in mathematical programming [9], etc are some examples of such areas. The other reason is that, the graph coloring problem has been shown to be computationally hard at a variety of levels: not only its decision problem variant is NP-complete [10], but also its approximate version is NP-hard [11].

These two reasons are important enough to justify the importance for new heuristics to solve graph coloring problem. A variety of heuristics and metaheuristics approaches have been proposed to produce optimal or near optimal colorings over the years by different researchers.

2. Heuristics and Metaheuristics for Graph Coloring

Sequential coloring approaches are the simplest heuristic methods. First, the vertices are sorted, and the top vertex is put in first color class. In order the remaining vertices are considered, and each nonadjacent vertex with respect to the already color class vertex, is placed in the color class number one. If no such class exists, then a new class is created. For the initial ordering, several different schemes have been used. The Largest First (LF) approach of Welsh and Powell [12] sorts the vertices by decreasing degree. The Smallest Last (SL) ordering of Matula [13] lists the smallest degree vertices in reverse order. Both LF and SL tend to color the higher degree nodes first. Although these methods are simple to implement and fast, but colorings they produce remain far from the optimal. By performing interchanges, the colorings produced by these sequential coloring algorithms can be improved. At each stage, if performing the interchange allows the current vertex to be colored without adding a new color, a previously colored vertex is switched to another class. As a result we get, Largest First with Interchange (LFI) and Smallest Last with Interchange (SLI) [13]. To reorder the nodes at each stage, other techniques have been developed. The next vertex to color is chosen based on its saturation degree — the number of color classes for which the vertex is adjacent to at least one vertex in that class, in DSATUR [14]. All the methods discussed above are choosing a vertex first and then assign an appropriate color. There are other approaches where each color is completed before introducing a new one. One such deterministic heuristics, Recursive Largest First (RLF), proposed by Leighton [2], is following this idea. In that method vertices of one color class got selected at a time. To improve the performance of simple heuristics sometimes, randomizations are also used. This idea is reflected in the work of Johnson, et al. [15], who developed the XRLF method. In that algorithm, several candidate classes are generated for each color, and the one with least degree in the remaining graph is chosen.

Metaheuristics, such as simulated annealing [16], tabu search [17], genetic algorithms [18] and neural networks [19], have also been applied to graph coloring. In most of the metaheuristics proposed, have the objective not to get stuck in local optima. They try to incorporate local search also which helps to move out from that situation in cost of deteriorating the value of the objective function. Application of simulated annealing to graph coloring was proposed by Chams, Hertz and Werra [20]. In their work they discussed, both pure simulated annealing and impure or hybrid version (combining XRLF and simulated annealing). In the hybrid method, XRLF is used to construct color classes until the pre specified number of vertices in the graph. Simulated annealing is then applied to color the remaining vertices. An extensive experimentation using several variants of simulated annealing on random graphs of varying size and density, was performed by Johnson, et al. [15]. A tabu search implementation for graph coloring was proposed by Hertz and Werra [21]. This tabu search gives good solutions and outperforms simulated annealing on different instances of random dense graphs. In their method, they attempted by changing k(number of colors) values to find a legal improved coloring. Another hybrid approach has been proposed by Hertz and Werra, where used tabu search within a sequential procedure to find large.

In the context of graph coloring evolutionary algorithms (EA) have also been applied by number of workers. In EA, a population of solutions is generated and using cooperation step and a self-adaptation step for number of iterations, it move towards betterment. In the cooperation step, solutions in the current population exchange information with the goal of producing new solutions that inherit good attributes. In the self-adaptation step, solutions modify their internal structure without interacting with other solutions in the population. In case of genetic algorithms (GA), iterations are referred to as generations, cooperation step is nothing but the crossover operators and the self-adaptation step consists of mutation. A hybrid method combining a simple descent method to achieve self-adaptation within the general framework of a genetic algorithm was proposed by Costa, Hertz and Dubuis [22]. In the descent method, proposed by Hertz and De Werra [21], by some means neighboring solution is generated from the solution. For assigning weights to edges, the objective function is modified. To avoid manipulating the same conflicting edges every time, in each generation, the weights are changed. The mutation operator generates a solution by a randomly chosen neighbor, with a given probability. A union crossover, originally designed by Costa [23] for a scheduling application, was adapted for graph coloring to implement the cooperation step. Another

evolutionary method for graph coloring was proposed by Fleurent and Ferland [24]. In their work they used a graph-adapted recombination operator. To evaluate the diversification of the solutions in the population an entropy measure was employed. All the solutions of the population are same for zero entropy value. This measure will not only act as stopping rules but also can influence parent selection. Members of the population are subject to local search, is also another feature of this work. Eiben, et al. [25] employed the 3-coloring problem to test several variants of an asexual evolutionary algorithm. In that work, they used an order-based representation and an adaptation mechanism. They finally applied this result for constraint satisfaction problems

Graph coloring problem has also been solved by neural networks. The first work was done by Dahl [26], and later continued by Jagota [27]. In neural network approach, k-coloring problem was mapped to a Hopfield network. This process is two step process. The first step is to reduce the problem to maximum independent set (MIS) problem, and in the second step, MIS will be mapped onto a Hopfield network. Jagota follows the approach of choosing an initial k value and gradually decreasing it in an attempt to find improved feasible colorings. The current k value is increased, if the algorithm fails to find a feasible coloring in one phase. The initial k is set to a sufficiently large value. As a result, the algorithm is guaranteed to yield a proper coloring. The algorithm was tested on a set of 30 graphs associated with the Second DIMACS Challenge [28]. The results were compared with the parallel procedure Hybrid of Lewandowski and Condon [29]. In all but 6 instances, Jagota's implementation is outperformed by Hybrid.

Kong, Wang, Andrew and Guo [30] presented a new hybrid genetic algorithm (GA), which embeds two kinds of local search algorithms - enumerative search and random search within the GA framework. In addition, they integrate a partition based encoding scheme with a specialized crossover operator into their GA method. Experimental results showed that their new algorithm outperforms the best-published results in terms of the quality of solutions and the computing time needed. Lim and Wang [31] applied various meta-heuristics including genetic algorithm, simulated annealing and tabu search for robust graph coloring problem (RGCP). Sivanandam, Sumathi and Hamsapriya [32], described a new permutation based representation of graph coloring problem. They employed a migration model of parallelism for GA with Message passing interface (MPI). They used three-crossover operators namely greedy partition crossover (GPX), Uniform independent set crossover (UISX), and Permutation-based crossover (PX) are used. The results on the benchmark graphs show that GPX performs well in terms of convergence and PX in terms of execution time. Kumar, Toley and Tiwary [33] considered a formulation of the bi-objective soft graph coloring problem, so as to simultaneously minimize the number of colors used as well as the number of edges that connect vertices of the same color. They used multi-objective evolutionary algorithms (MOEA) for good diversity and (local) convergence. They also adapted the single objective heuristics to yield a Pareto-front and observed that the quality of solutions obtained by MOEAs is much inferior. Recently, Ray, Pal, Bhattacharya and Kim [34] have used a new operator, called double point Guided Mutation operator with a special feature. An evolutionary algorithm with double point Guided Mutation for the Graph Coloring problem is proposed by them, which increased the performance level of simple GA dramatically.

We have provided a partial review of the graph coloring literature, focusing on different heuristic and hybrid approaches. For more information on the graph coloring problem and a more comprehensive bibliography, we refer Michael Trick's "Network Resources for Coloring a Graph" (http://mat.gsia.cmu.edu/COLOR/color.html), Joseph Culberson's "Graph Coloring Page" (http://webdocs.cs.ualberta.ca/~joe/Coloring/index.html).

3. Modified Simulated Annealing approach for Graph Coloring Problem

SA's major advantage over other methods is an ability to avoid becoming trapped in local minima. The algorithm employs a random search, which not only accepts changes that decrease the objective function f (assuming a minimization problem), but also some changes that increase it. The latter are accepted with a probability

$$p = \exp (-\delta f / T),$$

where δf is the increase in f and T is a control parameter, which by analogy with the original application is known as the system *"temperature*" irrespective of the objective function involved. The most important elements of the SA algorithm are:

- a representation of possible solution
- an operator of random changes in solutions to generate a neighbor solution

- a means of evaluating the solution's cost function and
- an annealing *schedule* - an initial temperature and rules for lowering it as the search progresses

3.1 Representation of Possible Solution & Cost

Here we represent the solution as a set of integers (1, 2, 3….n). These integers are nothing but the color of the nodes. The positions of these integers are the node numbers for which those particular colors have been assigned. An example of a solution (means coloration) for a graph of 5 vertices is shown below:

1 2 1 3 4

Here the nodes 1 and 3 have the same color 1. 2^{nd}, 3^{rd} and 4^{th} nodes have the colors 2, 3 and 4 respectively.

The cost function is nothing but the total number of colors used, i.e., distinct integers in the solution, e.g., the above solution has the cost 4.

The initial solution can be generated by randomly assigning colors to different nodes from n number of colors (n is the number of vertices). After generating initial solution randomly it may denote valid coloration or it may not. We have used a check operator to check its validity. But we have not discarded the solution if it's invalid. Instead, we use a repair operator *repair*, which converts the invalid solution to a valid one.

This *repair* checks whether two adjacent nodes have the same color or not. In that particular case, it replaces one such color by any randomly generated color except the earlier color. The repair operator is working in the following manner:

Repair operator (repair)

Step 1) Find out the vertex v_{mc} with maximum confliction in the solution S
Step 2) Repeat until S is valid
  Step 2.1) Change the color of v_{mc} using random color from the range (1 to m_color)
  Step 2.2) m_color ← m_color – 1

3.2 Operator of Random Changes in Solution to Generate a Neighbouring Solution

Here instead of using some operators to generate a neighboring solution randomly, we have used a special operator, which operates on a particular point of the solution randomly, to generate neighboring solution under a special guidance. The working of this operator is as follow:

Random change operator (rand_change_op)

Step 1) Choose a random position x in solution S
Step 2) Change the color of x^{th} position of S using a random color from the range (1 to m_color) to generate new solution S'
Step 3) Check validity of S'
Step 4) If S' is valid coloration
  Step 4.1) m_color ← m_color – 1
Step 5) Else
  Call *repair* operator to make it a valid coloration

When, we are working with the above two operators, we have considered a range of colors (1 to m_color). The initial value of this m_color is not n (number of vertices), instead we have considered as maximum_degree + 1.

Sometimes if there is no improvement of solution we have considered the bad solution with certain probability.

3.3 An Annealing Schedule

Initially we have considered the value of T as 10000 after trial and error method and at each step we have reduced the temperature by 0.01.

Now using all those operators let us write the complete algorithm for Modified Simulated Annealing (MSA) for Graph Coloring Problem (GCP).

3.4 Algorithm MSAGCP

INPUT: Adjacency matrix (n x n) of the graph with n vertices

Step 1) Initialize: T ← 10000
Step 2) Create one solution (S) using random coloration
Step 3) If the coloration is not a valid
call *repair* to make it valid
Step 4) Calculate cost of S, C(S)
Step 5) Repeat until exit_condition
Step 5.1) Apply *rand_change_op* operator on S
to obtain new solution S'
Step 5.2) Calculate cost of S', C(S')
Step 5.3) If C(S') < C(S)
Step 5.3.1) S ← S'
Step 5.4) Else
Step 5.4.1) delta ← C(S) – C(S')
Step 5.4.2) pb ← $e^{-delta/T}$
Step 5.4.3) If pb > *random [0, 1]*
Step 5.4.3.1) S ← S'
Step 5.4.4) T ← T – 0.01
Step 6) Output the coloration.

4. Computational Experiments

In this section, we present experimental results obtained by MSAGCP and make comparisons with the simple version of the same algorithms SAGCP. SAGCP is the simplified version or better to say the original version of simulated annealing, where the new operator *rand_change_op* is not there. Instead, just from the current solution changing the color of one position randomly, the neighboring solution was generated. In this section we present the results of our algorithm on some benchmark graphs given at http://mat.gsia.cmu.edu/COLOR04/. MSAGCP has been implemented in ANSI C and run on a Xeon 2.4GHz machine with 1GB of RAM running the Linux operating system. Initially we have considered the value of T as 10000 after trial and error method and at each step we have reduced the temperature by 0.01. The exit condition (exit_condition) of the algorithm is either the value of the temperature is 0 or the chromatic coloring has been generated. Both the two algorithms are tested on each of the benchmarks 10 times and the averages of ten execution times are given. In describing the instances, in the $\chi(G)$ column, the chromatic numbers are given, where it is not known, ? is given along with the still known best result.
In general, our observation is that, the SAGCP is taking a huge amount of time and failed to converge to optimal result in most of the cases. But due to the introduction of the new operator *rand_change_op* in MSAGCP, the convergence is faster and most of the cases it has generated optimal or near optimal result. For the queen series graphs, except those cases presented, in all other instances we got the result which is far from the optimal. Basically, the graphs which are hard in nature, our MSAGCP failed to generate optimal results. Table-1 shows the experimental results of the two algorithms and also the time taken by them.

Table 1. Comparative Results Between SAGCP and MSAGCP

Instances	V	E	$\chi(G)$	SAGCP	Time of SAGCP(S)	MSAGCP	Time of MSAGCP(S)
1-FullIns 5.col.b	282	3247	?/6	8	4078	6	1022
1-Insertions 5.col.b	202	1227	?/6	9	2098	7	796
2-FullIns 4.col.b	212	1621	?/6	6	4458	6	3454
2-Insertions 4.col.b	149	541	4	6	560	5	196
3-FullIns 4.col.b	405	3524	?/7	12	10706	9	2078
3-Insertions 4.col.b	281	1046	?/5	5	3086	5	658
4-FullIns 3.col.b	114	541	?/7	7	2286	7	127
4-Insertions 4.col.b	475	1795	?/5	7	20765	6	7835
5-FullIns 3.col.b	154	792	?/8	8	3395	8	517

anna.col.b	138	493	11	13	9855	11	2535
david.col.b	87	406	11	11	10672	11	2972
games120.col.b	120	638	9	10	2874	9	122
homer.col.b	561	1628	13	17	15438	12	4052
huck.col.b	74	301	11	11	1295	11	31
jean.col.b	80	254	10	11	520	10	45
miles1000.col.b	128	3216	42	50	25084	45	7702
miles1500.col.b	128	5198	73	80	15633	75	5885
miles750.col.b	128	2113	31	35	12767	31	2437
mug100 25.col.b	100	166	4	4	276	4	38
mug88 25.col.b	88	146	4	4	496	4	60
mulsol.i.1.col.b	197	3925	49	58	10439	52	2861
myciel6.col.b	95	755	7	8	12436	7	6683
queen5_5.col.b	25	160	5	7	186	5	40
queen7_7.col.b	49	476	7	9	276	7	188
queen9_9.col.b	81	2112	10	14	388	11	85
queen10_10.col.b	100	2940	?/11	17	798	12	121
queen11_11.col.b	121	3960	11	15	996	11	492
DSJC125.1.col.b	125	736	?/5	10	1094	6	201
DSJC500.1.col.b	500	12458	?/15	28	12865	17	6219
le450_5a.col.b	450	5714	5	11	6742	6	2768
le450_15a.col.b	450	8168	15	20	18342	17	10532
zeroin.i.2.col.b	211	3541	30	44	8874	30	2341

5. Conclusion and Future Scope

We have presented experimental results on some of the DIMACS benchmark graphs and compared the modified algorithm's performance with the simple one. Indeed, our MSAGCP able to find the best known results for most of the tested graphs. But in some instances, MSAGCP failed to generate optimal. That may due to the nature of the graph itself. To strengthen conclusions made about the power of the algorithm, it is worth to test it on some other classes of large graphs from the benchmark set as well as from the random graphs. The main purpose of this paper is to study simulated algorithm and its variation on graph coloring problem. In the future, we intend to refine MSAGCP and apply it to optimally color large complex graphs in reasonable time, as well as compare its performance with some other algorithms like genetic algorithm, ant colony optimization or may be some hybrid heuristics.

References

1. D.C. Wood, A technique for coloring a graph applicable to large scale time-tabling problems, *Computer Journal*, vol. 12, pp. 317-319, 1969.
2. F. T. Leighton, A graph coloring algorithm for large scheduling problems, *Journal of Research of the National Bureau of Standards*, vol. 84, no. 6, pp. 489-505, 1979.
3. F.C. Chow and J.L. Hennessy, Register allocation by priority based coloring, *Proceedings of the ACM SIGPLAN 84 symposium on compiler construction Newyork*, pp. 222-232, 1984.
4. M.R. Garey, D.S. Johnson and H.C. So., An application of graph coloring to printed circuit testing, *IEEE Transactions on circuits and systems*, vol. 23, pp. 591-599, 1976.
5. A. Gamst, Some lower bounds for class of frequency assignment problems, *IEEE Transactions on Vehicular Technology*, vol. 35, no. 1, pp. 8-14, 1986.
6. Micheli G.D., *SYNTHESIS AND OPTIMIZATION OF DIGITAL CIRCUITS.* McGraw-Hill, 1994.
7. S.S. Sarma, R. Mondal and A. Seth, Some sequential graph coloring algorithms for restricted channel routing, *INT. J. Electronics*, vol. 77, no. 1, pp. 81-93, 1985.
8. K. Stecke, Design, planning, scheduling and control problemsof flexible manufacturing, *Annals of Operations Research*, vol. 3, pp. 187-209, 1985.
9. T.F. Coleman and J.J. More, Estimation of sparse Jacobian Matrices and graph coloring problems, *SIAM. J. Numer. Anal.*, vol. 20, pp 187-209, 1983.
10. Garey, M.R. and D. S. Johnson, *COMPUTERS AND INTRACTABILITY : A GUIDE TO THE THEORY OF NP-COMPLETENESS*. Newyork, W. H. Freeman and Co., 1979.

11. Baase, S. and A.V. Gelder, *COMPUTER ALGORITHMS: INTRODUCTION TO DESIGN AND ANALYSIS*. Addison-Wesley, 1999.
12. D.J.A. Welsh and M.B. Powell, "An upper bound for the chromatic number of a graph and its application to timetabling problem," *Computer Journal*, vol. 10, pp. 85-86, 1967.
13. D.W. Matula, G. Marble and J.D. Isaacson, *GRAPH THEORY AND COMPUTING*. Newyork, Academic Press, 1972.
14. D. Brelez, New methods to color vertices of a graph, *Comm. ACM*, vol. 22, no. 4, pp.251-256, 1979.
15. D.S. Johnson, C.R. Aragon, L.A. McGeoch, and C. Schevon, Optimization by simulated annealing:an experimental evaluation; part II, graph coloring and number partitioning, *Operations Research*, vol. 39, no. 3, pp. 378-406, 1991.
16. S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, Optimization by simulated annealing, *Science*, vol. 220, pp. 671-680, 1983.
17. F. Glover and M. Laguna, *TABU SEARCH.* Kluwer Academic Publishers, 1997.
18. J.H. Holland, *DAPTATION IN NATURAL AND ARTIFICIAL SYSTEMS.* University of Michigan Press, 1975.
19. J.J. Hopfield, and D.W. Tank, Neural Computation of decisions in optimization Problems, *Biological Cybernetics,* vol. 52, pp. 141-152, 1985.
20. M. Chams, A. Hertz and D. Werra, Some experiments with simulated annealing for coloring graphs, *European Journal of Operational Research,* vol. 32, no. 2, pp. 260-266, 1987.
21. A. Hertz and D. Werra, Using tabu search techniques for graph coloring, *Computing,* vol. 39, no. 4, pp. 345-351, 1988.
22. D. Costa, A. Hertz and O. Dubuis, Embedding of a sequential procedure within an evolutionary algorithm for coloring problems in graphs, *Journal of Heuristics,* vol. 1, no. 1, pp. 105-128, 1995.
23. D. Costa, An evolutionary tabu search algorithm and the NHL scheduling problem, *INFOR*, vol. 33, no. 3, pp. 161-178, 1995.
24. C. Fleurent and J.A. Ferland, Genetic and hybrid algorithms for graph coloring, *Annals of Operations Research,* vol. 63, pp. 437-464, 1996.
25. A.E. Eiben, J.K. Hauw and J.I. Hemert,Graph coloring with adaptive evolutionary algorithms, *Technical Report, TR-96-1*, Leiden University, Netherlands, 1997.
26. E.D. Dahl, Neural Networks algorithms for an NP-complete problem: Map and graph coloring, *IEEE International Conference on Neural Networks,* vol. 3, pp. 113-120, 1987.
27. A. Jagota, An adaptive, multiple restarts neural network algorithm for graph coloring, *European Journal of Operational Research,* vol. 93, pp. 257-270, 1996.
28. M. Trick, The Second DIMACS Challenge, http://mat.gsia.cmu.edu/challenge.html, 1993.
29. G. Lewandowski and A. Condon, Experiments with parallel graph coloring heuristics, *Technical Report 1213*, University of Wisconsin, Madison, 1993.
30. Y. Kong, F. Wang, A. Lim and S. Guo, A New Hybrid Genetic Algorithm for the Robust Graph Coloring Problem, *Proceedings of Australian conference on artificial intelligence,* Perth, pp. 125-136, 2003.
31. A. Lim and F. Wang, Meta-heuristics for robust graph coloring problem, *Proceedings of 16th IEEE International Conference on Tools with Artificial Intelligence*, Florida, pp. 514-518, 2004.
32. S. N. Sivanandam, S. Sumathi and T. Hamsapriya, A hybrid parallel genetic algorithm approach for graph coloring, *International Journal of Knowledge-based and Intelligent Engineering Systems,* vol. 9, pp. 249 – 259, 2005.
33. R. Kumar, P. Toley and S. Tiwary, Enhancing solution quality of the biobjective graph coloring problem using hybridization of EA: biobjective graph coloring problem, *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, Atlanta, GA, USA, pp. 547-554, 2008.
34. B. Ray, A. J. Pal, D. Bhattacharyya, and T.H. Kim, An Efficient GA with Multipoint Guided Mutation for Graph Coloring Problems, *International Journal of Signal Processing, Image Processing and Pattern Recognition,*Vol. 3, No. 2, pp. 51-58,June, 2010