Memetic Algorithm to Solve Graph Coloring Problem

Hasin Al Rabat Chowdhury, Tasneem Farhat, and Mozammel H. A. Khan

***Abstract*—Graph coloring problem (GCP) is of great interest to the researchers in the area of soft computing. To solve GCP, we present a memetic algorithm (MA) that uses classical crossover operation as the main variation operator and a deterministic local improvement technique. For the first time in the literature, we use binary encoded chromosomes for GCP, which makes both the crossover and the local improvement easier. In the traditional evolutionary algorithm (EA) for GCP, *k*-coloring is used and the EA is run repeatedly until the lowest possible *k* is reached. In our MA, we start with the theoretical upper bound of chromatic number (maximum out-degree + 1) and in the process of evolution some of the colors are made unused to dynamically reduce the number of color. Thus, the solution is found in a single run of the MA reducing the total execution time in comparison to running *k*-coloring for several times. We experiment with 23 datasets taken from standard DIMACS benchmark and compare the result with several recent works. For all but one dataset, we obtain the minimum chromatic number stated in the DIMACS benchmark. For the remaining one dataset (queen10_10.col), we obtain better solution than others.**

***Index Terms*—Binary encoding, dynamicity, graph coloring, local improvement, memetic algorithm, metaheuristics.**

I. Introduction

Graph coloring problem (GCP) assigns different colors to the adjacent vertices of a graph using minimum number of colors. The GCP is illustrated with a simple graph in Fig. 1, where six colors are needed to color eleven vertices. In Fig. 1, the vertex number is shown within the circle and the color number is shown outside the circle. The GCP is a well known NP-hard problem [1]. The notable applications of GCP are seen in pattern recognition [2], map coloring [3], radio frequency assignment [4], bandwidth allocation [5], and timetable scheduling [6].

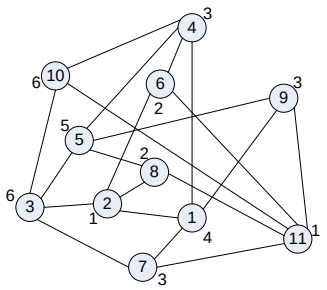

Fig. 1. Example of graph coloring.

Assume that a graph $G = (V, E)$ is to be colored with n numbers of colors. Our objective is to color all the vertices reducing n dynamically so that minimum chromatic number, denoted by $x(G)$, is found, that is, $n = x(G)$ is reached. It has been an established fact that metaheuristic approaches such as evolutionary algorithms (EA) are best suited for this class of optimization problem. In this paper, we propose a memetic algorithm (MA) that uses binary encoding of the chromosomes, classical crossover as the main variation operator, and a deterministic local improvement technique of the solution population.

Manuscript received February 10, 2013; revised April 15, 2013.

H. A. R. Chowdhury, T. Farhat, and M. H. A. Khan are with Department of Computer Science and Engineering, East West University, Aftabnagar, Dhaka 1212, Bangladesh (e-mail: hrcrabat@gmail.com, lvuplf@gmail.com, mhakhan@ewubd.edu).

II. Prior Work

One of the most recent works on GCP [7] combines wisdom of artificial crowds approach with the genetic algorithm (GA) and *k*-coloring approach is used. In this approach, multiple parent selection and multiple mutations based on the closeness of the solution to the global optima are used. The algorithm is run several times for several decreasing values of *k* and the minimum possible *k* value is taken as the minimum chromatic number. Another work used hierarchical parallel genetic algorithm for GCP [8], where the authors extended genetic algorithm with genetic modification operator introduced by them. A guided genetic algorithm for GCP called MSPGCA is reported in [9], where the authors finetuned the initial chromosomes using a simple genetic algorithm and then the deterministic MSPGCA algorithm is run to dynamically reduce the chromatic number. Tabu Search as a GCP solving approach was used in [10]. A hybrid immune algorithm is also applied in GCP [11]. All the above mentioned approaches used integer encoding for the chromosomes.

III. Proposed Chromosome Encoding Scheme

Integer encoding is used in the previous works on GCP. However, binary encoding has dominating success in the evolutionary computing for describing solution structure in depth. On the other hand, in a dynamic learning system, it is more feasible to understand learning progress and to improve quality of population at any point. Considering these facts, we use a binary encoding scheme for chromosomes in our MA, which allows us to implement the crossover operator easily and to deterministically improve the solution quality dynamically. The encoding scheme is illustrated in Fig. 2 for the undirected graph of Fig. 1. The used encoding is a two-dimensional array, where each row corresponds to a color and each column corresponds to a vertex. Let the *j*th vertex be colored using the *i*th color, then the (i, j)th element of the array will be 1 and the other elements will be 0. Thus, in a valid chromosome, every column must have a single 1 and a row will have one or more than one 1s placed on non-adjacent

vertices columns. A row may also have all 0s, in which case the color is not used in the solution. If a column has all 0s (the vertex is not colored) or more than one 1s (more than one color is assigned to that vertex), then the encoding is invalid. This situation may arise after crossover operation as discussed later, where two 1s may be present in one column. On the other hand, if a row has 1s in adjacent vertices columns (same color is assigned to adjacent vertices), then the encoding is invalid. This situation may arise during initialization as discoursed in the next section. When a chromosome becomes invalid, then the chromosome will be corrected as explained in Section IV of this paper. If a row has all 0s, then that color is not used in the vertex coloring. Thus, the number of rows having at least one 1 is the number of used colors and is used as the fitness function in our MA. As discussed in Section IV, this binary encoding technique will allow us to deterministically improve the quality of the solution.

colors ↓ ← vertices

	1	2	3	4	5	6	7	8	9	10	11
1	0	1	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	1	0	1	0	0	0
3	0	0	0	1	0	0	1	0	1	0	0
4	1	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	1	0	0	0	0	1	0
6	0	0	1	0	0	0	0	0	0	0	0

Fig. 2. Chromosome encoding of graph of Fig. 1.

IV. Proposed Memetic Algorithm for Graph Coloring

Our proposed memetic algorithm (MA) for graph coloring problem (GCP) is shown below and each step of the MA is discussed separately.

1) Memetic Algorithm for GCP ()
2) population = initialization ()
3) population = correction (population)
4) fitness = evaluation (population)
5) while (termination condition not reached) do
6) parents = parentSelection (population)
7) offsprings = crossover (parents)
8) offsprings = correction (offsprings)
9) offsprings = improvement (offsprings)
10) offsprings_fitness = evaluation (offsprings)
11) population = replacement (population, offsprings)
12) end
13) end

The initialization procedure of line 2 generates the chromosomes of the initial population. Chromosomes are initialized with $m + 1$ colors, where m is the maximum out-degree of the graph. The theoretical upper bound of the chromatic number is $m + 1$. The initial chromosome is generated by putting a single 1 under each column at a randomly selected row and filling up the remaining elements by 0s. During the initialization, a chromosome for the graph of Fig. 1 may be as shown in Fig. 3(a). In this chromosome, adjacent vertices 7 and 11 are colored by color 1 and adjacent vertices 4 and 6 are colored by color 2, which is an invalid chromosome. Therefore, correction is needed to make it a valid chromosome. For the correction procedure of line 3, one of the conflicting 1s is chosen randomly and then the chosen 1 is placed randomly in another row under the same column such that no conflict is created in that row. This technique is repeated until all color clusters become non-conflicting. The corrected chromosome corresponding to the invalid chromosome of Fig. 3 (a) is shown in Fig 3 (b). In the initial population of chromosomes, no duplicate chromosomes are allowed.

	1	2	3	4	5	6	7	8	9	10	11
1	0	1	0	0	0	0	1	0	0	0	1
2	0	0	0	1	0	1	0	1	0	0	0
3	0	0	0	0	0	0	0	0	1	0	0
4	1	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	1	0	0	0	0	1	0
6	0	0	1	0	0	0	0	0	0	0	0

(a)invalid

	1	2	3	4	5	6	7	8	9	10	11
1	0	1	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	1	0	1	0	0	0
3	0	0	0	1	0	0	1	0	1	0	0
4	1	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	1	0	0	0	0	1	0
6	0	0	1	0	0	0	0	0	0	0	0

(b)corrected

Fig. 3. Invalid chromosome during initialization and its correction.

The evaluation procedure of line 4 determines the fitness of each chromosome. The fitness of a chromosome is equal to the number of used colors in the chromosome.

In the termination condition of line 5, if both average and best fitness remain constant for a pre-determined number of generations then the algorithm is terminated.

crossover point →

	1	2	3	4	5	6	7	8	9	10	11
1	0	1	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	1	0	1	0	0	0
3	0	0	0	1	0	0	1	0	1	0	0
4	1	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	1	0	0	0	0	1	0
6	0	0	1	0	0	0	0	0	0	0	0

parent 1

crossover point →

	1	2	3	4	5	6	7	8	9	10	11
1	0	0	0	0	1	0	0	0	0	1	0
2	0	0	0	0	0	0	1	0	0	0	0
3	0	1	0	1	0	0	0	0	0	0	1
4	0	0	1	0	0	0	0	0	1	0	0
5	0	0	0	0	0	1	0	1	0	0	0
6	1	0	0	0	0	0	0	0	0	0	0

parent 2

	1	2	3	4	5	6	7	8	9	10	11
1	0	1	0	0	0	0	0	0	0	0	1
2	0	0	0	0	0	1	0	1	0	0	0
3	0	0	0	1	0	0	1	0	1	0	0
4	0	0	1	0	0	0	0	0	1	0	0
5	0	0	0	0	0	1	0	1	0	0	0
6	1	0	0	0	0	0	0	0	0	0	0

offspring 1

	1	2	3	4	5	6	7	8	9	10	11
1	0	0	0	0	1	0	0	0	0	1	0
2	0	0	0	0	0	0	1	0	0	0	0
3	0	1	0	1	0	0	0	0	0	0	1
4	1	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	1	0	0	0	0	1	0
6	0	0	1	0	0	0	0	0	0	0	0

offspring 2

Fig. 4. The crossover operation.

In the parent selection procedure of line 6, two parents are

randomly selected from the population. In the crossover procedure of line 7, a crossover point is randomly selected and the rows above the crossover point are interchanged between the two parents. Two parent chromosomes for the graph of Fig. 1 and the generated offsprings are shown in Fig. 4. In our MA, the crossover operation is applied with high probability. The crossover operation may produce invalid offsprings. Two possible problems may occur – 1) a column may have two 1s or 2) a column may have all 0s. In Fig. 4, both the generated offsprings are invalid and both have the two possible problems. In the correction procedure of line 8, the invalid offsprings are corrected. For case 1), one of the two 1s is randomly deleted. For the case 2), a 1 is inserted at a randomly selected row among used color clusters so that no conflict is created at that row. If such a used color row is not available, then a 1 is inserted at a randomly selected row among unused color clusters. An invalid offspring and its corrected version are shown in Fig. 5.

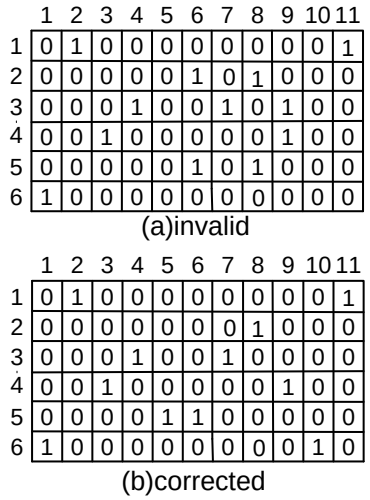

Fig. 5. Correction of invalid offspring.

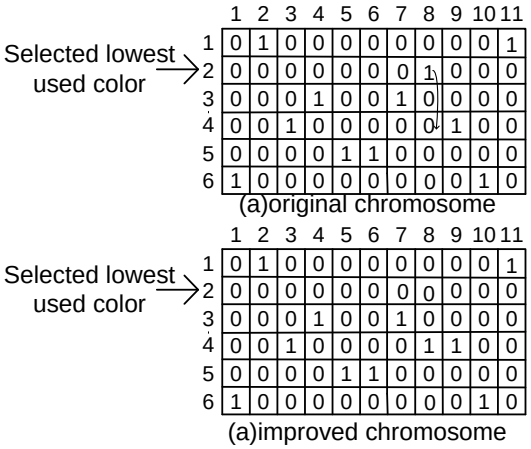

Fig. 6. Local improvement of a chromosome.

Improvement procedure of line 9 deterministically improves the quality of the offsprings. For each offsprings, a lowest used color cluster is selected randomly. Then 1s of that selected color cluster are moved to other rows of the used color clusters so that no conflict is created at that row. If any 1 of the selected color cluster cannot be moved to any of the rows of the used color clusters, then it is left in the original color cluster. This local improvement procedure makes a color unused improving the fitness of the offspring. The improvement procedure is applied with a low probability. One possible chromosome for the graph of Fig. 1 and its improved version is shown in Fig. 6. In Fig. 6 (a), colors 2 is the lowest used. The 1 of the 2nd row can be moved to row 2, 3, 4, or 6 without conflict. It is moved to row 4 to produce the improved chromosome of Fig. 6 (b). The fitness of the chromosome of Fig. 6 (a) is six and that of the Fig. 6 (b) is five.

V. Experimental Results

Datasets used to test our memetic algorithm for GCP are taken from Center for Discrete Mathematics and Theoretical Computer Science (DIMACS) benchmarking graph collection [12]. Instances ending in .col are in DIMACS standard format. Instances in .col.b are in compressed format. We have used datasets ending with .col extension. The top of the dataset heading resembling "*p* edge 11 20" means that graph has 11 vertices and 20 edges, where p denotes vertices. After that number of lines like "*e* 1 2" represent connection between two edges.

The proposed memetic algorithm for GCP is written in Java utilizing JDK 1.7 64bit, random numbers are generated using commons-math-2.0 and tests were run on a desktop PC having following configuration:

CPU: Intel Core2Quad 2.66 GHz
Memory: 4 GB DDR3 1333MHz
Operating System: Windows 7 64-bit

TABLE I: Comparison of Tested Datasets Results

Dataset	$\|V\|$	$\|E\|$	$x(G)$	[7]	[8]	[9]	MA
myciel2.col	5	5	3	-	-	-	3
myciel3.col	11	20	4	4	4	-	4
myciel4.col	23	71	5	5	5	-	5
myciel5.col	47	236	6	6	6	-	6
myciel6.col	95	755	7	-	-	-	7
myciel7.col	191	2360	8	-	8	-	8
games120.col	120	638	9	9	9	9	9
huck.col	74	301	11	11	11	11	11
jean.col	80	254	10	10	10	10	10
david.col	87	406	11	11	11	11	11
queen5_5.col	25	160	5	5	5	5	5
queen6_6.col	36	290	7	7	8	8	7
queen7_7.col	49	476	7	7	8	7	7
queen10_10.col	**100**	**2940**	**?**	**-**	**15**	**14**	**13**
miles250.col	128	387	8	8	8	-	8
miles500.col	128	1170	20	-	-	-	20
miles750.col	128	4226	31	-	-	31	31
miles1000.col	128	3216	42	42	42	42	42
miles1500.col	128	5198	73	-	73	73	73
anna.col	138	493	11	11	11	11	11
homer.col	561	1629	13	13	13	13	13
mulsol.i.1.col	197	3925	49	-	49	49	49
zeroin.i.1.col	211	4100	49	-	-	-	49

We experimented with 23 datasets. The tested datasets are heterogeneous consisting of big graph like homer.col having 561 vertices, highly dense graph like miles1500.col, highly complex graph like queen10_10.col, and even simple graphs. Results of our algorithm are compared with those of three

most recent works in Table I. The table has eight columns, where dataset is the name of the DIMACS dataset, $|V|$ and $|E|$ are the number of vertices and edges of the datasets respectively from [12], $x(G)$ denotes expected chromatic number as found in [12], the next three columns show the found chromatic number of three recent works, and the last column shows the chromatic number found in our experiments denoted by MA. A "?" mark in the fourth column indicates that no expected chromatic number is reported in [12]. A "–" mark in fifth to seventh columns means that no result is reported in the corresponding work. For 22 datasets, except queen10_10.col dataset, we found the expected chromatic number as mentioned in [12]. For queen10_10.col dataset, no expected chromatic number is mentioned in [12]. For this dataset, the found chromatic number of [8] and [9] are 15 and 14, respectively. The work of [7] did not report any result for this dataset. In our experiment, we found 13 as the chromatic number of this dataset, which is a major achievement of our algorithm over the previous works.

In the previous works such as that of [7] used k-coloring technique and the algorithm is run for several decreasing k values. The minimum possible k value is taken as the found chromatic number. Unlike this technique, our memetic algorithm starts with $m + 1$ colors, where m is the maximum out-degree of the graph, and dynamically reduces the used colors to find out the chromatic number. The value $m + 1$ is the theoretical upper bound of chromatic number of a graph. Thus, our memetic algorithm

For GCP starts with theoretical upper bound of chromatic number and dynamically reduces the used colors to reach at the minimum chromatic number. Fig. 7 shows the average fitness (number of used color) and minimum fitness over successive generation for queen5_5 dataset indicating the dynamicity of our algorithm.

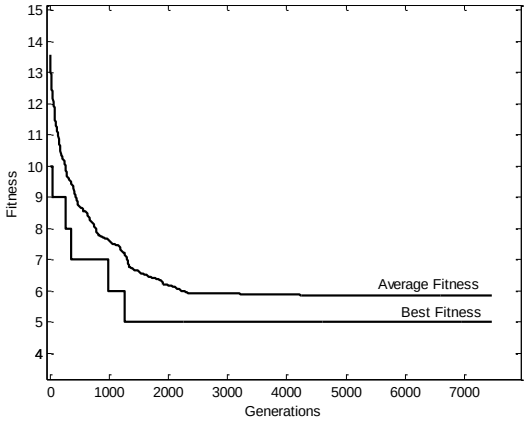

Fig. 7. Average and best (minimum) fitness over generation for queen5_5 dataset.

In our experiments, we found that crossover probability of 0.7 performs better for all datasets. However, the probability of local improvement depends on graph density. If the number of edges is less than 10, than an improvement probability of 0.1 works better, otherwise a probability of 0.25 works better. The termination condition also depends on the graph complexity. As stated earlier, if both the average fitness and the best fitness do not change for a specified number of generations, then the algorithm is terminated. This number of generations depends on the graph complexity. For queen5_5.col dataset this number was 5000 and for myciel7.col dataset it was 40000.

VI. Conclusion

In this paper, we propose a memetic algorithm (MA) for graph coloring problem (GCP). Unlike the previous works, we use a binary encoding scheme for the first time for GCP. The main variation operator of our MA is the classical crossover operator of the genetic algorithm (GA). That means the population of the solutions is updated mainly using crossover operator. We select two parents randomly and apply the crossover operator with a high probability. Due to the nature of the encoding, the generated offsprings may become invalid and in that case the offsprings are corrected to valid solutions. Then a deterministic improvement technique is applied on the corrected offsprings with low probability to locally improve the solution quality. If the generated offspring is better than the worst solution of the population and if it is also not duplicate of any other solution of the population, then the worst solution is replaced by the offspring. The binary encoding makes the local improvement procedure easy. The combination of the genetic operation and the deterministic improvement makes the algorithm a MA.

We start with $m + 1$ colors, where m is the maximum out-degree of the graph. The number $m + 1$ is the upper bound of chromatic number. That means, we start with upper bound of the chromatic number and the MA dynamically reduces the chromatic number to the possible minimum chromatic number in a single run.

We experiment with 23 DIMACS dataset [12]. For 22 datasets, except queen10_10.col dataset, we found expected chromatic number as stated in [12]. For queen10_10.col dataset, no expected chromatic number is stated. In our experiment we found this number to be 13, whereas this number found in [8] and [9] are 15 and 14, respectively. Thus, our MA outperforms the previous works for a very complex dataset. Moreover, unlike the previous techniques, our MA finds the minimum chromatic number in a single run reducing the total run time significantly.

References

[1] M. Kubale, "Graph colorings," *American Mathematical Society*, 2004.
[2] C. W. K. Chen and D. Y. Y. Yun, "Unifying graph-matching problem with a practical solution," in *Proc. International Conf. on Systems, Signals, Control, Computers*, September 1998.
[3] B. H. Gwee, M. H. Lim, and J. S. Ho, "Solving four-colouring map problem using genetic algorithm," in *Proc. Artificial Neural Networks and Expert Systems*, 1993.
[4] W. K. Hale, "Frequency assignment: theory and applications," in *Proc. IEEE*, 1980, vol. 12, pp. 1497-1514.
[5] A. Gamst, "Some lower bounds for class of frequency assignment problems," *IEEE Trans. on Vehicular Technology*, vol. 35, no. 1, pp. 8-14, 1986.
[6] N. K. Cauvery, "Timetable scheduling using graph coloring," *International Journal of P2P Network Trends and Technology*, vol. 1, issue 2, pp. 57-62, 2011.
[7] M. M. Hindi and R. V. Yampolskiy, "Genetic algorithm applied to the graph coloring problem," in *Proc. 23rd Midwest Artificial Intelligence and Cognitive Science Conf.*, April 2012, pp. 61-66.
[8] R. Abbasian and M. Mouhoub, "An efficient hierarchical parallel genetic algorithm for graph coloring problem," in *Proc. 13th annual conf. on Genetic and evolutionary computation*, 2011, pp. 521-528.

[9] B. Ray, A. J. Pal, D. Bhattacharyya, and T. 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, 2010.
[10] A. Lim, Y. Zhu, Q. Lou, and B. Rodrigues, "Heuristic methods for graph coloring problems," in *Proc. Symposium on Applied Computing*, 2005.
[11] V. Cutello, G. Nicosia, and M. Pavone, "A hybrid immune algorithm with information gain for the graph coloring problem," in *Proc. GECCO 2003*, pp. 171-182, 2003.
[12] M. Trick. (2013, March 1). Graph coloring instances. *Michael Trick's Operations Research Page*. [Online]. Available: http://mat.gsia.cmu.edu/COLOR/instances.html

Hasin Al Rabat Chowdhury was born in Khulna, Bangladesh. He is a final year student of Computer Science and Engineering at East West University, Aftabnagar, Dhaka 1212, Bangladesh. His research interests include Evolutionary Algorithms and Machine Learning.

Tasneem Farhat was born in Dhaka, Bangladesh. She is a final year student of Computer Science and Engineering at East West University, Aftabnagar, Dhaka 1212, Bangladesh. Her research interest is in Evolutionary Algorithms.

Mozammel H. A. Khan was born in Kushtia, Bangladesh. He received B. Sc. Engg. degree in Electrical and Electronic Engineering, M. Sc. Engg. degree in Computer Engineering, and Ph.D. degree in Computer Science and Engineering in 1984, 1986, and 1998, respectively, all from Bangladesh University of Engineering and Technology (BUET), Dhaka, Bangladesh.

He has served as faculty member at Bangladesh Institute of Technology (BIT), Rajshahi, Bangladesh and Khulna University, Khulna, Bangladesh. He is currently a full Professor of Department of Computer Science and Engineering at East West University, Aftabnagar, Dhaka 1212, Bangladesh. His research interests include Logic Synthesis, Reversible Logic, Multiple-Valued Logic, Quantum Computing, and Evolutionary Algorithms. He is an author or co-author of about 75 papers published in international and national journals and conferences. He also authored two textbooks.

H. A. Khan is a senior member of IEEE and a life fellow of Institution of Engineers, Bangladesh.