Scientific Journal of Impact Factor(SJIF): 3.134

e-ISSN(O): 2348-4470
p-ISSN(P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 1,Issue 12, December -2014

GENETIC ALGORITHM AND MEMETIC ALGORITHM ON GRAPH COLORING

Priyanshi Barod[1], Varsharani Hawanna[2], Vandana Jagtap[3]

[1]priyanshii.barod@gmail.com,[2]hawanna.varsha@gmail.com,[3]vandana.jagtap@mitpune.edu.in.
Computer Department, MAEER's MIT, Pune, India.

Abstract— *Graph coloring problem (GCP) is of great interest to the researchers in the area of soft computing. This paper compares the formulation and results of two recent evolutionary-based algorithms used for Graph Coloring : genetic algorithm, Memetic algorithm. This paper also Gives basic understanding of Genetic Algorithm , Memetic algorithm with Operators used in GA and MA*

Keywords—*Graph Coloring; Genetic Algorithm; Memetic Algorithm; Crossword; Mutation; Selection; Integer Encoding; Binary Encoding*

I. INTRODUCTION

The Graph Coloring Problem (GCP) is a well-known NP complete problem. The term graph coloring usually refers to vertex coloring .Given a number of vertices, which form a connected graph, the objective is to color each vertex such that if two vertices are connected in the graph (i.e. adjacent) they will be colored with different colors. Moreover, the number of different colors that can be used to color the vertices is limited and a secondary objective is to find the minimum number of different colors needed to color a certain graph without violating the adjacency constraint [1].

Applications of graph coloring:

- Map coloring
- Scheduling
- Radio Frequency Assignment
- Register allocation
- Pattern Matching
- Sudoku
- Computer Network Security
- Global Positioning System (GPS)

II. OPERATORS IN GA AND MA

1. **SELECTION**:

The selection process is based on fitness. Chromosomes that are evaluated with higher values (fitter) will most likely be selected to reproduce, whereas, those with lower values will be discarded. The fittest chromosomes may be selected several times, however, the number of chromosomes selected to reproduce is equal to the population size, therefore, keeping the size constant for every generation. This phase has an element of randomness just like the survival of organisms in nature [5].

2. **CROSSOVER**:

Crossover is the process of combining the bits of one chromosome with those of another. This is to create an offspring for the next generation that inherits traits of both parents. Crossover randomly chooses a locus and exchanges the subsequences before and after that locus between two chromosomes to create two offspring.

For example, consider the following parents and a single crossover point at position 3
Parent 1 1 0 0 | 0 1 1 1
Parent 2 1 1 1 | 1 0 0 0

Offspring1 1 0 0 1 0 0 0
Offspring 2 1 1 1 0 1 1 1 [5].

3. **MUTATION**:

Mutation is performed after crossover to prevent all solutions in the population to fall into a local optimum of solved problem. Mutation changes the new offspring by flipping bits from 1 to 0 or from 0 to 1. Mutation can occur at each bit position in the string with some probability, usually very small (e.g. 0.001).

For example, consider the following chromosome with mutation point at position 2:

Not mutated chromosome: 1 *0* 0 0 1 1 1

mutated: 1 *1* 0 0 1 1 1

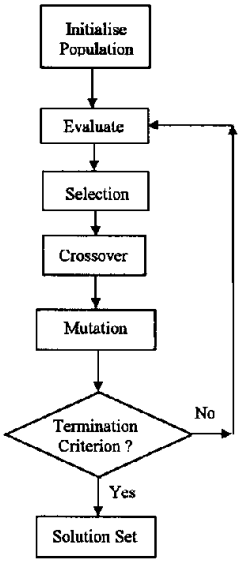
The 0 at position 2 flips to 1 after mutation.

Mutation probability is the percentage of total number of genes in the population. The mutation probability controls the probability with which new genes or memes are introduced into the population for trial. If the probability is too low, many genes that would have been useful are never tried out, while if it is too high, there will be much random perturbation, the offspring will start losing their resemblance to the parents and the algorithm will lose the ability to learn from the history of the search.[5]

4. **LOCAL SEARCH:**

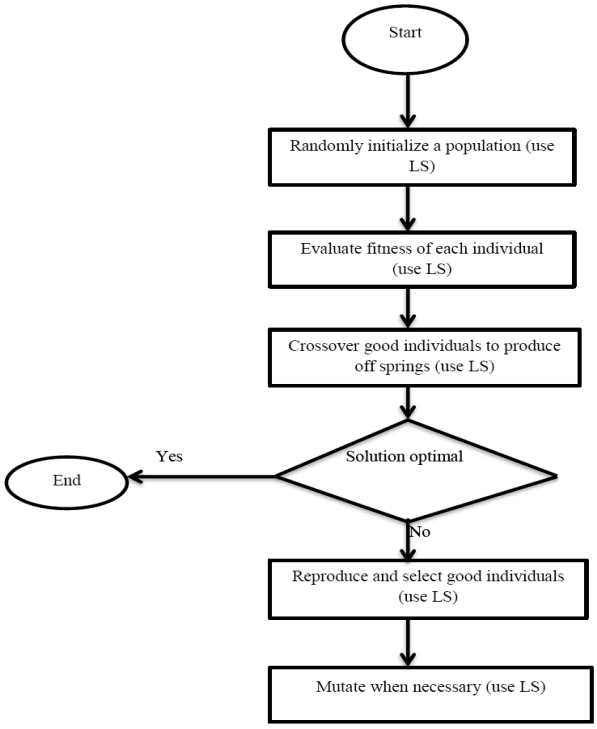
local search is a heuristic for solving optimization problems. They have the ability that when provided with knowledge, they exploit the entire search space to find the best solution [2].

III. GENETIC ALGORITHM



Fig(a) : Flow Chart of Genetic Algorithm

IV. MEMETIC ALGORITHM



Fig(b): Flow Chart Of Memetic Algorithm

V. GENETIC ALGORITHM FOR GRAPH COLORING

Genetic algorithm consists of a parent selection method, a crossover method and a mutation.

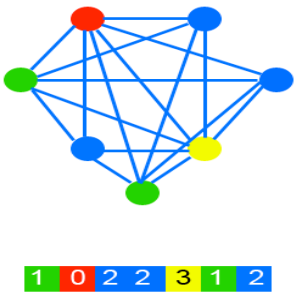
The general algorithm is:

Algorithm1: General Genetic Algorithm

```
population = randomly generated chromosomes;
while (terminating condition is not reached)
{
gaRun();
}
// a single run of a genetic algorithm
function gaRun() {
parents = getParents();
child = crossover(parents);
child = mutate(child);
population.add(child);
        }
```

The chromosome representation of the GCP is simply an array with the length set to the number of vertices in the graph. Each cell in the array is assigned a value from 0 to the number of colors – 1. The adjacencies between the vertices are represented by an adjacency matrix of dimensions $n \times n$ where n: the number of vertices

The ultimate goal when solving GCPs is to reach a solution where no two adjacent vertices have the same color. Therefore, the GA process continues until it either finds a solution (i.e. 0 conflicts) or the algorithm has been run for the predefined number of generations[1]



Fig(C) : Chromosome representation of a colored connected graph

The GA uses two different parent selection methods, a single crossover method and two different mutation methods. Which of the parent selection and mutation methods ends up selected depends on the state of the population and how close it is to finding a solution. The parent selection, crossover and mutation methods are outlined as follows:

Algorithm2: parentSelection1:
tempParents = two randomly selected chromosomes from the population;
parent1 = the fitter of tempParents;
tempParents = two randomly selected chromosomes from the population;
parent2 = the fitter of tempParents;
return parent1, parent2;

Algorithm3: parentSelection2:
parent1 = the top performing chromosome;
parent2 = the top performing chromosome;
return parent1, parent2;

Algorithm4: crossover
crosspoint = random point along a chromosome;
child = colors up to and including crosspoint from parent 1 + colors after crosspoint to the end of the chromosome from parent2;
return child;

Algorithm5: mutation:
for each(vertex in chromosome) {
if (vertex has the same color as an adjacent vertex) {
adjacentColors = all adjacent colors;
validColors = allColors – adjacentColors;
newColor = random color from validColors;
chromosome.setColor(vertex, newColor)
}
}return chromosome;

A bad edge is defined as an edge connecting two vertices that have the same color. The number of bad edges is the fitness score for any chromosome. As mentioned above, the alteration between the two different parent selection and mutation methods depends on the best fitness. If the best fitness is greater than 4 then parentSelection1 and mutation are used.

The overall genetic algorithm in this approach is a generational genetic algorithm. The population size is kept constant at all times and with each generation the bottom performing half of the population is removed and new randomly generated chromosomes are added.[1]

VI. MEMETIC ALGORITHM FOR GRAPH COLORING

Assume that a graph $G = (V, E)$ is to be colored with n numbers of colors

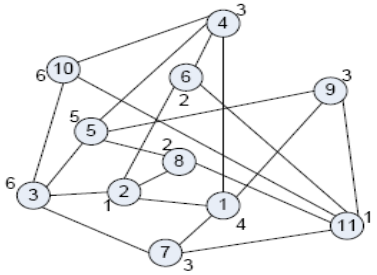


Fig (D) : Example of graph coloring

The used encoding is a two-dimensional array, where each row corresponds to a color and each column corresponds to a vertex. Let the jth vertex be colored using the ith 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

And in Invalid chromosome situation where 1)A row may also have all 0s, in which case the color is not used in the solution.2) If a column has all 0s or more than one 1s3) if a row has 1s in adjacent vertices columns

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.

colors ↓ 1 2 3 4 5 6 7 8 9 10 11 ←vertices

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 (E): Chromosome encoding of graph of Fig. (D)

MemeticAlgorithm(MA)Steps[5]for graph coloring Problem is:

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. (D) may be as shown in Fig. (F-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. (F-a) is shown in Fig. (F-b). In the initial population of chromosomes, no duplicate chromosomes are allowed[5].

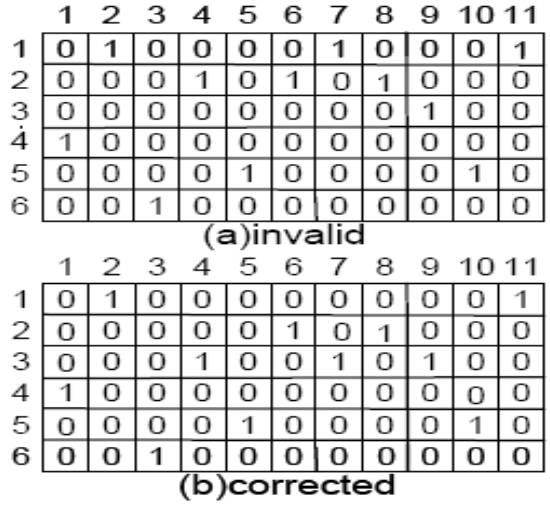


Fig (F): Invalid chromosome during initialization and its correction.

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.

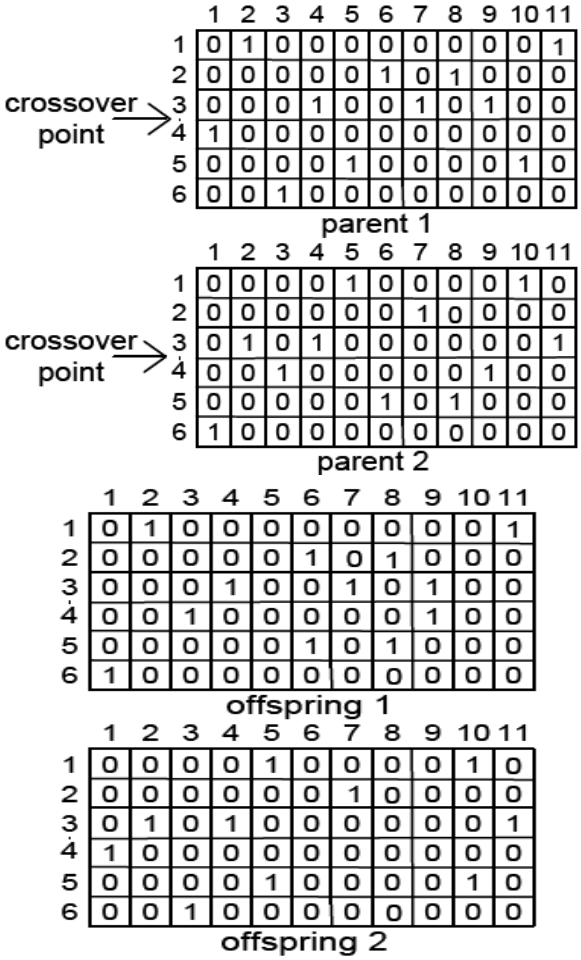


Fig. (G) : The crossover operation

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. (G). 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. (G),
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.(H).

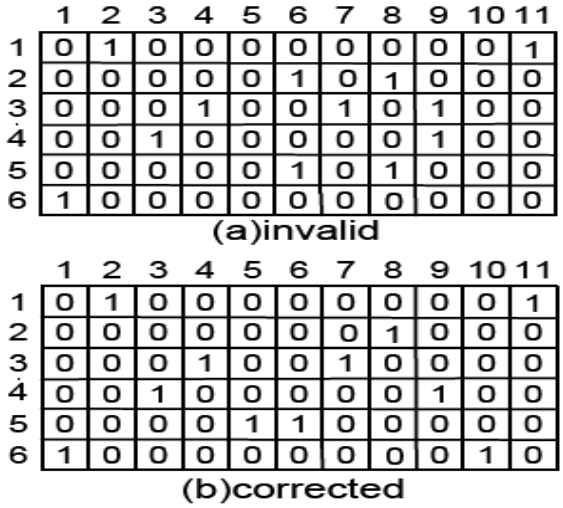


Fig. (H) : Correction of invalid offspring

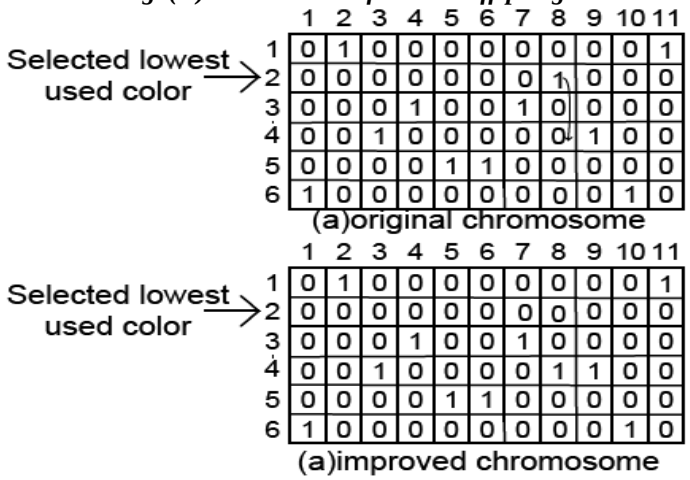


Fig. (I) : Local improvement of a chromosome

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.(I). In Fig. (I- 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.(I-b). The fitness of the chromosome of Fig.(I- a) is six and that of the Fig.(I-b) is five [5].

VII. COMPARISON BETWEEN GENETIC AND MEMETIC ALGORITHM OF GRAPH COLORING

In Genetic algorithm of Graph Coloring 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. But This approaches used Integer Encoding for the chromosomes.

Where as in Memetic Algorithm Graph Coloring Approach We use Binary Encoding .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.

In MA 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. 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.

VIII. CONCLUSIONS

By reviewing Two methods above a major difference between the algorithms is that GA used Integer Encoding and MA used Binary Encoding for the chromosomes.

Another difference is that the , 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.

GA used Global Optima for solution where as MA used local optima for solution.

However, it is safe to state that memetic algorithm is more efficient and better than genetic algorithm since it is guaranteed to converge to an optimal result within a reasonable amount of iterations regardless of the initial population size.

REFERENCES

[1] Musa M. Hindi and Roman V. Yampolskiy, "*Genetic Algorithm Applied to the Graph Coloring Proble"*,Computer Engineering and Computer Science J.B. Speed School of Engineering Louisville, Kentucky .

[2] Memetic Algorithm to Solve Graph Coloring Problem, *International Journal of Computer Theory and Engineering, Vol. 5, No. 6, December 2013*.

[3] Memetic Algorithm: Introduction, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 8, August 2013 .

[4] Pablo Moscato, Carlos Cotta,"A GENTLE INTRODUCTION TO MEMETIC ALGORITHMS".

[5] H. A. Sanusi, A. Zubair , R. O. Oladele, Comparative Assessment of Genetic and Memetic Algorithms, Journal of Emerging Trends in Computing and Information Sciences, VOL. 2, NO. 10, October 2011.

[6] Emad Elbeltagia,*, Tarek Hegazyb,1, Donald Griersonb,2, Comparison among five evolutionary-based optimization algorithms, Advanced Engineering Informatics 19 (2005) 43–53, Received 15 October 2004; revised 9 January 2005; accepted 19 January 2005.

[7] Petrica C. Pop1, Bin Hu2, and G□unther R. Raidl2, A Memetic Algorithm for the Partition Graph Coloring Problem.

