JCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper **Volume-2, Issue-5** **E-ISSN: 2347-2693**

Efficacy of Different Strategies in Graph Coloring with Parallel Genetic Algorithms

Fardin Esmaeeli Sangari[1*], Mehrdad Nabahat[2]
[1*,2] *Sama technical and vocational training college, Islamic Azad University, Urmia branch, Urmia, Iran*
[1*] fardin_e_s@yahoo.com, [2] *m_nteach2009@yahoo.com*

www.ijcseonline.org

Received: 01/04/2014 Revised: 16 /04/2014 Accepted: 12/05/ 2014 Published: 31 /05/2014

***Abstract*— In this paper a new parallel genetic algorithm is proposed to observe efficacy of different strategies for k-graph coloring problem. In the algorithm we have applied a coarse-grained model of parallelism, along with two new algorithms for crossover and mutation are represented: FCX and Fmm. these algorithms compared with CEX's First Fit and Transposition mutation operators. In our experiments, we observed that different strategies what role have in finding solutions. In computer simulations of PGA we used DIMACS benchmark.**

***Keywords*— Graph Coloring Problem, Migration Model, Migration Strategy, Parallel Genetic Algorithm**

I. Introduction

Genetic Algorithms are the most recognized form of evolutionary algorithms. They are being applied successfully to find acceptable solutions to problems in business, engineering, and science .GAs are generally able to find good solutions in reasonable amount of time, but as they are applied to harder and bigger problems there is an increase in the time required to find adequate solutions. Fortunately, GAs work with a population of independent solutions, which makes it feasible to distribute the computational load among several processors. Indeed, some may say that GAs are "embarrassingly parallel" programs, and that it is trivial to make fast parallel GAs [1].

We have applied parallel genetic algorithm to represent a new algorithm for GCP (graph coloring problem). Graph coloring problem (GCP) belongs to the class of NP–hard combinatorial optimizations problems. GCP is defined for an undirected graph as a problem of assignment of available colors to graph vertices providing that adjacent vertices are assigned different colors and the number of colors is minimal [2].

A coloring using at most k colors is called a (proper) **k-coloring**. The smallest number of colors needed to color a graph G is called its **chromatic number**, $\chi(G)$. A graph that can be assigned a (proper) k-coloring is **k-colorable**, and it is **k-chromatic** if its chromatic number is exactly k. A subset of vertices assigned to the same color is called a color class; every such class forms an independent set. Thus, a k-coloring is the same as a partition of the vertex set into k independent sets, and the terms k-partite and k-colorable have the same meaning.

Problem (k-GCP): Let G=(V,E) be a connected graph where V is set of vertices ($|V|=n$), $E\subseteq V\times V$ is set of edges and $k\in N^+$ is a number of colors used. Our target is to find: Ψ: $V\rightarrow 1,2,3,\ldots,k$ ($u,v\in V$ and $[u,v]\in E$) subject that $\Psi(v)\neq\Psi(u)$.

There are already different algorithms for crossover operator, including SPPX, UISX, GPX and CEX [2] which are applicable to GCP, however as shown in [2] CEX is more efficient than others finding correct solution.

In this paper two new breeding operators for coloring chromosomes are proposed: FCX (Find Conflict Crossover) that is an extended version of CEX (Conflict Elimination Crossover) [2], and FMM (Find Maximum mutation) in which conflicts are substituted by randomly chosen color. The graph coloring instances for feeding our approach are taken from [7, 8, 9], these instances are presented in DIMACS standard format.

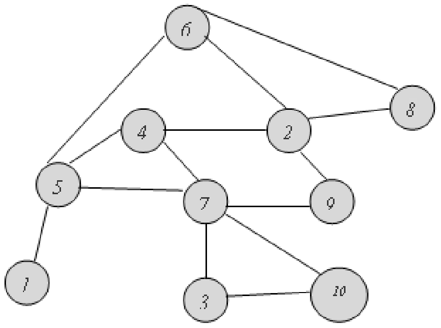
II. Genetic Operators for k-GCP

In order to achieve better results, some parameters and specifications need to be explained. Encoding in genomes, methods for Genetic Operators, selection function, defining a proper fitness function, and Migration Topology have considerable influence on how much the results are optimal. In this section we try to explain some of these parameters in our PGA.

III. Chromosome Representation

The first decision which has direct impact on performance is chromosome structure. *Assignment representation* and *partition representation* are widely used for k-GCP. In assignment representation, each genome is represented as a vertex assignment array which contains the colors assigned to the vertices. For instance, value j at index i means that color C_j is assigned to vertex V_i. On the other hand, in partition representation, each genome is represented as a partition of graph vertices. Every member of a subset in the partition has a same color.

Corresponding Author: *fardin_e_s@yahoo.com*



Partition representation: {1,5,9} {2,10} {4,8} {3,6,7}
Assignment representation: <1,2,4,3,1,4,4,3,1,2>

Fig.1. examples of both representation methods

For example, if value *i* is a member of subset *j* in partition representation, it means that color C_j is assigned to vertex V_i in the graph [3].

In our algorithm assignment representation is used. Fig.1 shows examples of both representation methods for given graph.

IV. Find Conflict Crossover

The proposed algorithm works in two phase. In first phase when two selected chromosomes have unequal conflicts, CEX algorithm is applied [2]. In second phase when two selected chromosome have equal conflicts, our algorithm is applied.

However, as our experiments show just in 40% of crossover operations CEX is called and rest of operations use FCX. (See Fig.2 and Fig.3)

```
Procedure coloring-crossover:
Begin
        Create first population randomly;
        Evaluate fitness of chromosomes;
        Repeat
                    ⎧ Select two parents;
                    ⎪ If num.Ofconflict [par.1] =
                    ⎪         num.Ofconflict [par.2]
                    ⎪     Call FCX;
        Crossover   ⎨ Else
                    ⎪     Call CEX;
                    ⎪ End if;
                    ⎪ If (threshold)
                    ⎩      Call FMM;
                      If (convergence is not acceptable)
                           Call FMM;
        Until condition is satisfied;
  End.
```

Fig.2. the main body of algorithm

```
Procedure:  FCX (par1, par2, child1, child2)
Begin
  Chromosomes_length=V;
    For (i=1; i<=chromosomes_length; i++)
        Exchange the conflict places in par1 and par2
        And replaces in child1 and child2;
    End for;
  End.
```

Fig.3. the recombination operator FCX

Example 1

Two parents represent different 4-coloring of a graph with 10 vertices i.e. sequences par1=<**1**,2,**4**,3,**1**,4,**4**,3,1,2> and par2=<1,**3**,4,**3**,2,4,**1**,2,**1**,2>. Vertices with color conflicts are marked bold and underlined. As example1 shows both par1 and par2 have 6 vertices with feasible colors and 4 vertices with color conflicts. Replacing the vertices with color conflicts in par1 by vertices with color conflicts in other parent, we obtain the following two chromosomes: child1=<3,2,3,3,**1**,4,**1**,3,**1**,2> and child2=<1,1,4,4,2,4,1,2,4,2>. It is observed that obtained chromosomes represent now two different 4-coloring of the given graph and the number of color conflicts is now reduced to 3 in child1 and reduced to 0 in child2 chromosomes (No conflicts in child2). (See Fig.1)

V. Conflict Elimination Crossover

In conflict–based crossovers for GCP, an assignment representation of coloring is used and offspring try to copy conflict–free colors from their parents.

Conflict Elimination Crossover (CEX) reveals some similarity to the classical crossover. Each parental chromosome is partitioned into two blocks. The first block consists of conflict–free nodes while the second block is built of the remaining nodes that break the coloring rules. The last block in both chromosomes is then replaced by corresponding colors taken from the other parent. This recombination scheme provides inheritance of all good properties of one parent and gives the second parent a chance to reduce the number of existing conflicts. However, if a chromosome represents a feasible coloring the recombination mechanism is not working. Therefore, the recombination must be combined with an efficient mutation mechanism. The operator CEX is almost as simple and easy to implement as the classical crossover (see Fig.4) [2].

```
Procedure: CEX (p,r,s,t)
Begin
    s = r;
    t = p;
    Copy conflict-free vertices $V_{cf}^{p}$ from p to s;
    Copy conflict-free vertices $V_{cf}^{r}$ from r to t;
End
```

Fig.*4. The recombination operator CEX.*

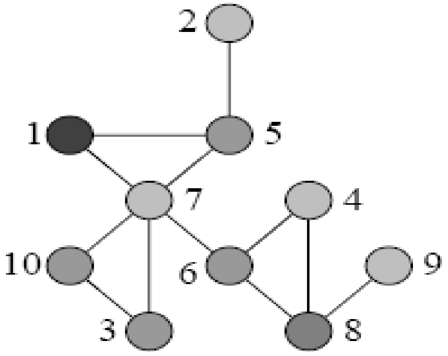


Fig.5. An illustration of CEX crossover (see Example 2)

Example 2

Two parents represent different 5-colorings of a graph with 10 vertices i.e. sequences p=<5,2,**3**,**4**,1,**4**,2,5,1,**3**>, and r=<1,4,**5**,2,3,3,**5**,4,2,**5**>. Vertices with color conflicts are marked by bold fonts. Thus, the chromosome p has 6 vertices with feasible colors and 4 vertices with color conflicts while the chromosome r has 7 vertices with feasible colors and 3 vertices with color conflicts. Replacing the vertices with color conflicts by vertices taken from the other parent we obtain the following two chromosomes: s=<5,2,**5**,2,1,3,2,5,1,**5**> and t=<1,4,**3**,2,3,3,2,4,2,**3**>. It is observed that obtained chromosomes represent now two different 4-colorings of the given and the number of color conflicts is now reduced to 2 in each chromosome [4]. (See Fig.5)

VI. Mutation Operators

Transposition is a classical type of mutation that exchange colors of two randomly selected vertices in the assignment representation. The second mutation operator called First Fit is designed for colorings in partition representation and is well suited for GCP. In the First Fit mutation one block of the partition is selected at random and propagate this block vertices to others using heuristic First Fit such that assignment of vertices to other blocks remain them conflict–free. Thus, as a result of the First Fit mutation the color assignment is partially rearranged and the number of partition blocks is often reduced by one [4].

But we proposed a new algorithm for mutation operator that is called Fmm (Find maximum mutation) in which algorithm selects a chromosome randomly then finds node with biggest conflict on chromosome and tries to change that with random color. (See Fig. 5)

```
Procedure: Fmm (chromosomes)
Begin
    Select parent randomly;
    X=Find index with biggest conflict in parent;
    Change the parent's Xth index color with- random color;
    Child=parent;
Return child;
End
```

Fig 6. The recombination operator Fmm

VII. Evaluation And Selection Function

We have used function (1) for evaluating chromosome's quality:

$$\mathrm{Pen}(c) = \sum_{(u,v)\in E} \mathrm{conflict}(u,v) + 1 , \qquad (1)$$

- C is a solution,
- Pen(c) is evaluation function for a solution (Note: bigger value for Pen(c) means less quality solution)
- Conflict is a penalty function for pairs of vertices connected by an edge (u,v)∈E: Conflict(u,v) = 2 when c(u) = c(v), and Conflict(u,v) = 0, otherwise.

For selection function, the tournament selection has been applied. This method has been used widely in PGA. However in our selection method each chromosome allowed to attend in competition just once, so that all chromosomes in population have equal chance of being selected. Equation (2) is our fitness function.

$$Fitness = \frac{1}{Pen(c)} \qquad (2)$$

VIII. Migration Model Of Parallel Genetic Algorithm

There are several ways to parallelize GA, and the parallel GA (PGA) has been developed and successfully applied to optimize practical problems. According to the nature of the population structure and recombination mechanisms used, PGA can be classified into four categories: single-population master–slave PGA, coarse-grained PGA, fine-grained PGA and hierarchical hybrids [5].

The coarse-grained parallel GA is very popular and widely used. In a coarse-grained PGA the entire population is divided into several sub-populations. Each sub-population runs a conventional GA independently and concurrently on their "island". After several epochs, some individuals migrate from one sub- population to another according to a migration topology (see Fig.6).

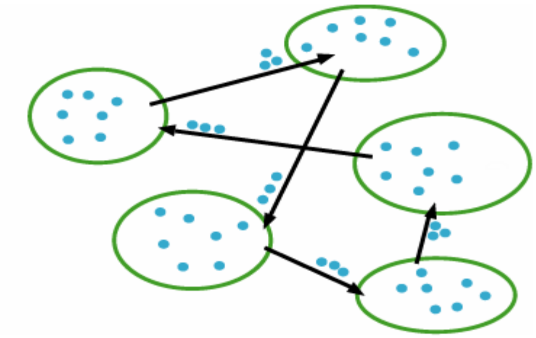


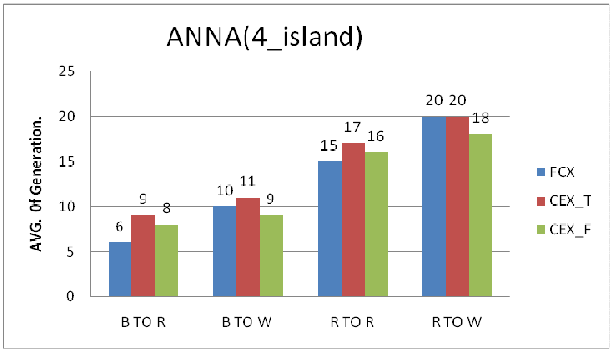
Fig.7. Migration topology

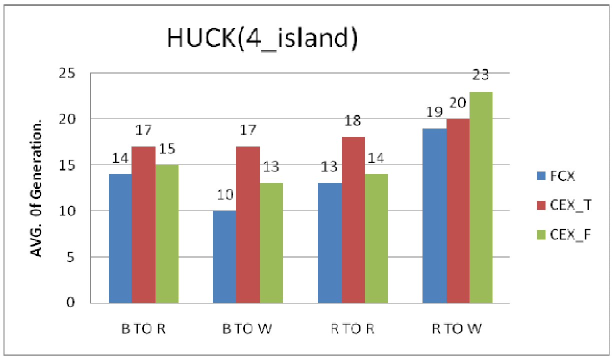
The individuals that emigrate from an island may be selected at random, or they may be selected among the best individuals in an island. Similarly, migrants may replace random individuals at the receiving island, or they may replace the worst individuals [6].

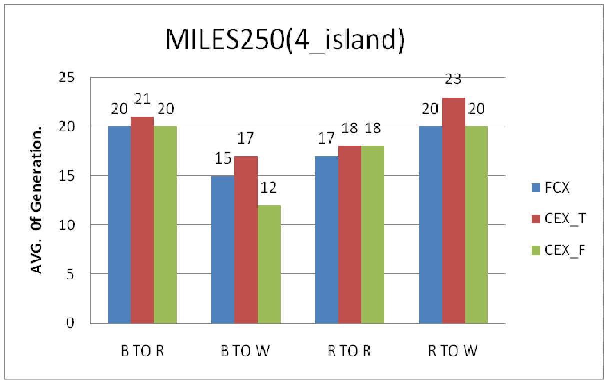
IX. Experimental Results

We have used four DIMACS graph coloring benchmarks from [7] to apply our experiments. In our PGA coarse-grained model is implemented. In our program different migration strategies are applied. Prepared results are computed using Intel Pentium 2.2GHz with 512MB memory. The following parameters were applied in our algorithm:

Population size=90, number of generations=2000, number of Island=3, number of migrants=5, migration interval=15, crossover probability=0.6, mutation probability for First Fit and Transposition=0.1. Experiments include comparing performance of both FCX and CEX crossover operators; it also examines efficiency of First Fit, Transposition and Fmm mutation operators. However, Fmm's mutation probability is dynamically changes based on algorithm convergence. All experiments were repeated 10 times. Fig.8 Indicative the generation numbers at the four different strategies that the first solutions have reached (Note in Fig.7, X-axis shows average of generation iteration and Y-axis shows different strategies).







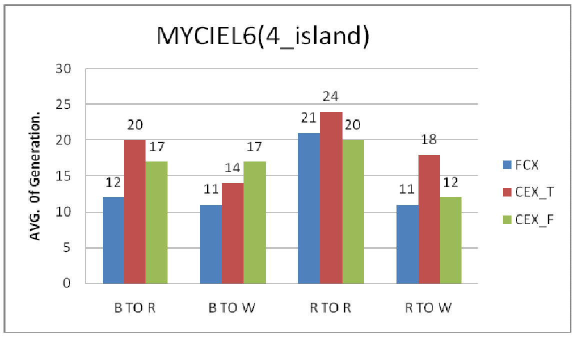


Fig.8. Number of generations to find the first solution in different strategies

X. Conclusion

In this paper, a new PGA algorithm is proposed to observe efficacy of different strategies for k-graph coloring problem (k-GCP). A new crossover and mutation algorithm called FCX and Fmm designed in a way that successfully could find more correct answers in reasonable time for different strategies. As described before, applying FCX in case of equal conflicts makes convergence more rapid, in addition, using coarse-grained model for implementation along with a proper migration fraction and interval influences our results.

References

[1] Erick Cantu-Paz David E. Goldberg, "Efficient parallel genetic algorithms: theory and practice*", J. Computer Methods in Applied Mechanic and Engineering*. 186, 2000, pp.221-238

[2] Zbigniew Kokosinski, Marcin Kolodziej, Krzysztof Kwarciany, "Parallel Genetic Algorithm for Graph Coloring Problem", *ICCS 2004*, LNCS 3036, pp. 215–222

[3] Saeed Amizadeh, Farzad Rastegar, Caro Lucas, "Incorporating Heuristics In Evolutionary Optimization", *J. Intelligent Information Technology Computing*, Vol.1, No.2, 2006, pp. 259-270

[4] Zbigniew Kokosinski, Krzysztof Kwarciany, Marcin Kolodziej, "Efficient Graph Coloring With Parallel Genetic Algorithms*", J. Computing and Informatics,* Vol. 24, 2005, pp. 1001-1025

[5] Z.G. Wang, Y.S. Wong, M. Rahman, "Development of a parallel optimization method based on genetic simulated annealing algorithm*", J. Parallel Computing,* Vol. 31 ,2005, pp. 839–857

[6] Erick Cantu-Paz, "Migration Policies, takeover Times in parallel genetic algorithms*", Department of computer science and Illinois Genetic Algorithm Laboratory,* 1999

[7] http://mat.gsia.cmu.edu/COLOR/instances.html

[8] ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/

[9] http://mat.gsia.cmu.edu/COLORING03/