A Novel Presentation of Graph Coloring Problems Based on Parallel Genetic Algorithm

Saideh Naderi, Masoud Jabbarian, Vahid Sattari Naeini

Abstract — In coloring graph idealization the minimum number of required colors for graphic coloring is determined in a way that no contiguous summit have the same color this number is called chromatic graph number. We should decide if then, a color for a given integral number M , so we use that number with no contiguous summits of the same color there have been presented several algorithms for decision and idealization manners so for such as: reverse counting method ,space mood counting method and etc…that don't follow multi statement time.

Here by in this paper we present suitable solutions for this problem by genetic algorithm. In order to evaluate the performance of our new approach, we have conducted several experiments on GCP instances taken from the well known DIMACS Website. The results show that the proposed approach has a high performance in time and quality of the solution returned in solving graph coloring instances taken from DIMACS website. The quality of the solution is measured here by comparing the returned solution with the optimal one.

Index Terms: Graph Coloring Problems (GCPs), Parallel Genetic Algorithms (PGAS), NP-hard, chromosome.

I. INTRODUCTION

The Graph Coloring Problem (GCP) is a well-known NP-complete problem. Graph coloring includes both vertex coloring usually refers to vertex coloring rather than edge 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. That number for a given graph (G) is known as the Chromatic Number ($\chi(G)$) (1).

If k = {1, 2, 3...} and P(G, k) is the number of possible solutions for coloring the graph G with k colors, then:

$$\chi(G) = \min(k: P(G, k) > 0) \quad (1)$$

Graph coloring problems are very interesting from the theoretical standpoint since they are a class of NP- complete problems that also belong to Constraint Satisfaction Problems (CSPs)

The practical applications of Graph Coloring Problems include but are not limited to:

1. Map coloring (2)
2. Scheduling (3)
3. Radio Frequency Assignment (4)
4. Register allocation (5)
5. Pattern Matching
6. Sudoku

In this paper we demonstrate the use of genetic algorithms in solving the graph-coloring problem while strictly adhering to the usage of no more than the number of colors equal to the chromatic index to color the various test graphs.

II. THE REVIEW OF PREVIOUS WORKS

A great deal of research has been done to tackle the theoretical aspects of the Graph Coloring Problem in terms of its generalization as a Constraint Satisfaction Problem (8). The problem's various applications and solutions have been discussed in detail in Porumbel's paper (9).

Evolutionary computation and parameter control has been detailed in a number of papers including ones by Back, Hammel, and Schwefel (7) as well as work by Eiben, Hinterding and Michalewicz (10). Srinivas and Patnaik examined crossover and mutation probabilities for optimizing genetic algorithm performance (6). Genetic algorithms and evolutionary approaches have been used extensively in solutions for the Graph Coloring Problem and its applications (7,1).

The concept of utilizing a crowd of individuals for solving NP complete problems has also been the topic of various papers. Most notably the Wisdom of Crowds concept has been used in solving the Traveling Salesman Problem as well as the Minimum Spanning Tree Problem (2,10).

There are generally three approaches to solve the Graph coloring problems (GCP) (8, 9). The first one consists in directly minimizing the number of colors by working in the legal colors space of the problem. In the second approach, the number of colors is fixed and no conflict is allowed, thus, some vertices might not be colored.

The objective here is to maximize the number of colored vertices (3,5). The third approach consists of first choosing a number of colors K, and then iteratively try to minimize the number of conflicts for the candidate K .

Whenever a solution with zero conflicts has been found, K is decremented by one and the procedure continues until were each a K where the number of conflicts cannot be equal to zero. As a result, the last legal K will be returned as the best solution (4).

Manuscript Received July, 2013.

Saideh naderi, Department of Computer Engineering, Kerman Branch, Islamic Azad University, Kerman, Iran.

Masoud jabbarian, Department of Computer Engineering, Kerman Branch, Islamic Azad University, Kerman, Iran.

Vahid Sattari Naeini, Department of Computer Engineering, University of Kerman, 7616914111 Kerman, Iran.

Genetic Algorithms (Gas) are categorized as global search heuristics and are generally able to find good solutions in reasonable amount of time.

A genetic algorithm is an iterative method that evolves a population of elements, which are encoding strings representing a possible selection for a given real-world problem. GAs have been successfully applied to problems that are difficult to solve using conventional techniques or procedures. Common application areas are: scheduling problems (1), graph coloring problems (4), traveling salesman problem, optimization problems (5), network routing problems for circuit-switched networks, financial marketing, bio-informatics and genomics. Gas are easily parallelized algorithms. Parallel genetic algorithms (PGAs) are parallel implementations of GAs that can provide gain in terms of computational performance and scalability. There exist two major kinds of parallelism in genetic algorithms:

In the computation of the fitness functions of individuals and 2- in the application of genetic operators (selection, crossover and mutation). An overview of theoretical advances, computing issues, applications and future trends in parallel genetic algorithms can be found in (6,9) A more detailed discussion of parallel genetic algorithms can be found in (2).

A general framework for studying and analyzing PGAs was proposed by Alba and Troya (5,10).

Recently, GAs can be applied for supervised and unsupervised data mining sessions. For data mining purposes we use population individuals as elements defined by attributes and values. These elements or individuals represent candidate production rules.

III. GRAPH COLORING PROBLEM

Given an undirected graph $G = (V, E)$ and color class $C = (1,2, , k0)$, the GCP is to color each node in such a way that no two nodes connected by an edge are colored with the same color with a minimum number (the chromatic number). The minimum number of colors (or the minimum color classes) needed to color a specific graph G , is called the chromatic number $\chi(G)$. A coloring which used k colors is called a k − coloring and can be regarded as the partition of the vertices of the graph to distinct color classes. Given a specific coloring assignment, if two adjacent vertices have the same color we say that these vertices are in conflict and call the connecting edge between them, a conflicting edge. Suppose V = {v1, v2 ,… , vn} be the node set and E = {e1, e2 ,… , em} be the edge set of graph G , we denote :

A(G) = (aij)n×n as the adjacent matrix of the nodes.

$$A_{ij}=\begin{cases}1 & (vi, vj) \in E \\ 0 & (vi, vj) \notin E\end{cases} \quad (2)$$

Hertz and De Werra (2) proposed a simple coding for GCP and a natural 1-exchange neighborhood.

In addition, they introduced a pre-processing technique which removes some independent sets from a graph leading to a reduced residual graph. Their algorithm, combined with this preprocessing, has produced excellent results on random graphs. Fleurent and Ferland investigated a tabu algorithm and in particular a hybrid algorithm combining genetic algorithm and tabu search (4).

They replaced random mutation by tabu search and develop a specialized crossover operator based on confliction nodes (adjacent nodes having the same color).

The graph coloring problem can be regarded as the following optimization problem: to determine a partition of V in a minimum number (the chromatic number $\chi(G)$) of color classes such no conflicting edges existed.

Therefore, the GCP is to optimize the two objectives: the number of color classed and the number of the conflicting edges. A optimal coloring of G is a k − coloring with the chromatic number $\chi(G)$. Therefore, the GCP can be formulated as the following bi-objective programming problem:

$$\text{Min } F(x) = (k(x), p(x)) \quad (3)$$

where k(x) is the number of color classes of individual x ,and p(x) is simply the number of the conflicting edges of individual x .

$$p(x) = \sum_{i=1}^{m} q(e_i), q(e_i) = \begin{cases}1 & e_i \in E, x_i = x_j \\ 0 & otherwise\end{cases} \quad (4)$$

Optimizing k(x) means searching the chromatic number, and optimizing p(x) means searching for the k(x) − coloring assignment. Thus simultaneously optimizing both k(x) and p(x) means looking for not only a k(x) − coloring assignment, but also a $\chi(G)$ – coloring assignment. The goal of the optimization process here is to minimize F(x) until k(x) = $\chi(G)$, p(x) = 0 , which corresponds to the optimal solution of the GCP.

IV. PARALLEL GENETIC ALGORITHMS

Genetic Algorithms (GAs) (9) are evolutionary algorithms based on the idea of natural selection and evolution. GAs have been successfully applied to a wide variety of problems.

In GAs, there is a population of potential solutions called individuals.

The GA performs different genetic operations on the population, until the given stopping criteria are met. The Parallel Genetic Algorithm (PGA) is an extension of the GA. The well-known advantage of PGAs is their ability to facilitate different subpopulations to evolve in diverse directions simultaneously (10). It is shown that PGAs speedup the search process and can produce high quality solutions on complex problems (1, 6, 8).

There are mainly three different types of PGA (3). First, Master-Slave PGA in which, there is only one single population and the population is divided into fractions. Each fraction is assigned to one slave process on which genetic operations are performed. Second, Multi-Population PGA (also called Island PGA) that contains a number of sub populations, which can occasionally exchange individuals.

The exchange of individuals is called migration. Migration is controlled using several parameters. Multi-population PGAs are also known as Island PGA, since they resemble the “island model” in population genetics that considers relatively isolated demes. Finally, the Fine-Grained PGA which consists of only one single population, that is designed to run on closely linked massively parallel processing systems.

A. Models of parallel genetic algorithms

There are many models of parallelism in evolutionary algorithms: master{slave PGA, migration based PGA,

PGA with over lapin subpopulations, population learning algorithm, hybrid models etc. (7,5).

The above models are characterized by the following criteria:

- number of populations : one, many;
- population types : disjoint, overlapping;
- population topologies : various graph models;
- interaction model : isolation, migration, diusion;
- recombination, evaluation of individuals, selection: distributed/local, centralized/global;
- synchronization on iteration level: synchronous/ asynchronous algorithm.

The most common models of PGA are:

- master-slave : one global population, global genetic operations, fitness functions computed by slave processors);
- massively parallel (cellular): static overlapping subpopulations with a local structure, local genetic operations and evaluation;
- migration (with island as a sub model): static disjoint subpopulations/islands, local genetic operations and migration;
- hybrid : combination of one model on the upper level and other model on the lower level (the speedup achieved in hybrid models is equal to product of level speed ups).

V. PROPOSED APPROACH

Defect of the old algorithms is convergence speed and not finding the general EXTERMEM so power full and efficient algorithms like natural idealization methods have been created.

Some of these algorithms are: genetic algorithm ,comparing heating, idealization of pieces group, used idealization , and finally evolutionary algorithms. All of these new algorithms produce new places in the searching space by doing some functions ,and moves to the idealized places gradually.

this method is based on a kind of intelligent searching in a big but limited space ;and unlike the old ones ,these algorithms don't need any derivatives computation so then won't be any limitation for the cost unconnected function and also connected derivatives.

We present a new method for coloring the graph below(fig 1) with four colors in a way that the contiguous graphs are not of the same color .in order to solve the graph with genetic algorithm we need to follow the below steps:

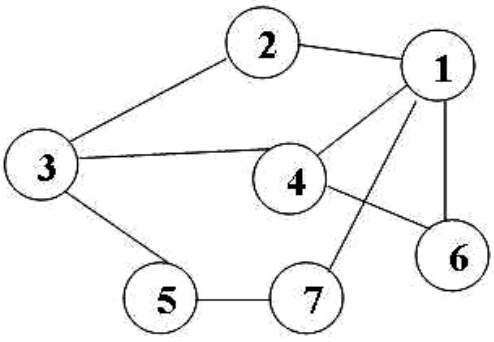

Fig.1: Graph coloring

Step1: turn it in to genetic software.

Step2: Then we consider gene for each chromosome with the graph head numbers.

Step3: In this problem, graph has got seven heads, so for each chromosome we consider seven genes. for each gene we specialize a random color . The function of Insert Matrix Color is as follows:

```
Algorithm1:The function of Insert Matrix Color ( ) begin
readln (txtcolor);
L:=lenght(txtcolor);
Matrixcolor =Array[L];
For all i:=0 to L do
begin
matrixcolor(i):=copy(txtcolor, i+1,1);
end;
```

Step4: The oriinal population is a number of chromosome which is regarded as 100 here in this paper we examine seven chromosome of produced population .In figure.2, people with the chromosome of this population are shown.

```
Algorithm2:The function of Create Population( )
begin
Randomize;
pop=random(100);
readln (vertis);
populate=Array[pop,vertis];
for all i:=0 to pop do
begin
for j:=0 to vertis do
begin
fork if j= vertis then
populate(i,j):=0
else
populate(i,j):=matrixcolor(r);
end;
end;
end;
```

Step 5:

if you reach to the chromosome or chromosomes ,which can be the problem answer , algorithm will end in the original population after random coloring of the chromosomes ,algorithm examines them to know which chromosome can be the answer .for example here we examine chromosomes of (6).

```
Algorithm3:Thefunction of Calculate Fitness()
begin
answer=Array[pop,vertis];
k:=(vertis*vertis)-vertis;
for i:=0 to pop do
begin
forall j:=0 to vertis do
begin
fork for p:=0 to vertis do
begin
if (j<>p) then
begin
if (matrixvertis (j,p)=0) then s++ ;
if (matrixcolor(j,p)=1) and (populate(i,p)<>populate(i,j))
then s++;
end;
end;
end;
populate(i,vertis):=str(s);
```

```
if s=k then
begin
flag=flag+1;
;x:=i
forall n:=x to i do
begin
forall e:=0 to vertis do
begin
answer(n,e):=populate(n,e);
end;
end;
end;
s:=0;
end;
```

by examining seven chromosomes of the original population we find out that chromosome number 1 and 5 can be chosen as the answer, because their random received chromosomes in this example are the correct answer. Here, by using chromosome 1 we color the graph as below. For example if we choose chromosome 2as the answer there would be failure in the problem because the heads 5 and 3 one contiguous with red color(figure .3).

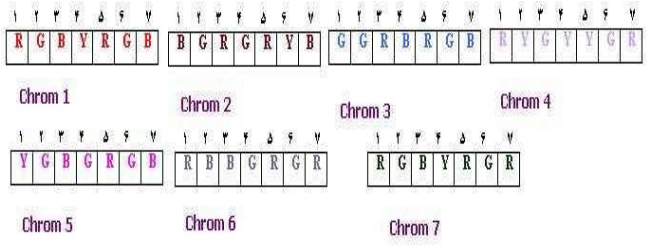

Fig.2: chromosome display and specifying random color.

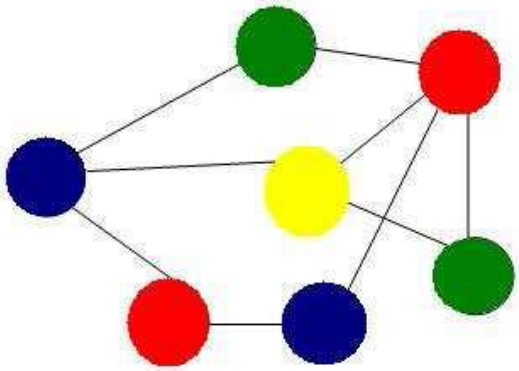

Fig.3: The graph resulted from coloring

A. Chromosome cutting and generating new child

Below ,then is an example of producing new child, Just because we use this form in the project . cutting cost in this project is regarded :cross =0.07, so all the parents are not able to generate ,which is agreed with the generating rules also ,because it may happen in the nature too. Here we use single spot method in order to produce child. We divide two chromosomes in half and produce the new child by mixing the results .

```
Algorithm4:The function of cross( )
begin
newpop:=pop*0.7;
mutation:=newpop*0.01;
newpop:=newpop+mutation;
mid:=(vertis/2);
newpopulate=Array[newpop+mutation,vertis];
a:=pop-newpop;
while (i<pop-a) do
begin
for j:=0 to vertis do
begin
newpopulate(i,j):=populate(i+1,j);
newpopulate(i+1,j):=populate(i,j);
end;
i+=2;
End;
End;
```

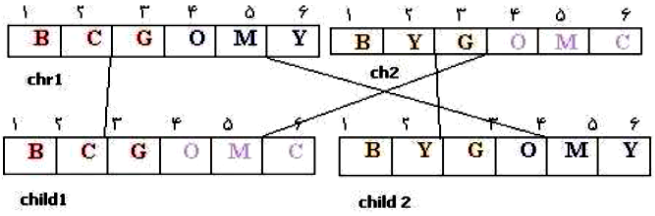

Fig.4 : producing new child

B. Chromosome mutation method

mutation cost in this paper is 0.01 in order to have mutation of one chromosome ,we choose two random spots and change their places. This process is shown below through an example.

```
Algorithm5:The function of Mutation()
begin
if mutation<>0
begin
while (i<mutation) do
begin
r:=rnd_next(0,pop);
L:=rnd_next(0,L);
u:=rnd_next(0,L);
temp:=populate(r,u);
populate(r,u):=populate(r,L);
populate(r,L):=temp;
forall j:=0 to vertis do
begin
mutationpop(i,j):=populate(r,j);

end;
i:=i+1;
end;
```

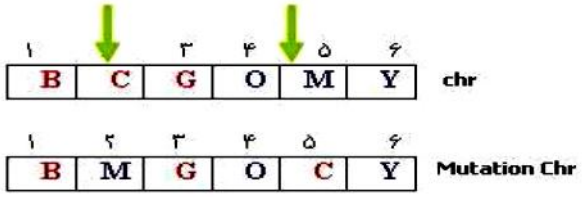

Fig. 5: Generate new child

VI. EXPERIMENT RESULTS AND ANALYSIS

In decision making problems like returning to the past the answer is a collection of indexes which are chosen from a wider collection of choices;

---

[1]http://mat.gsia.cmu.edu/COLOR03/

or it's a chain of decisions that find a final answer for the given problem any way the collection of possible choices or decisions is too vast and the time of related algorithms is not sensible Returning to the past method doesn't have a positive effect on reduction of time ; but it try s to approve the time of algorithm for the length of the small data by reducing of examinable manners.

genetic algorithm uses natural selection method of draw in to find a suitable formula for predicting and adjusting of the sample actually genetic algorithm is a method of searching for finding an ideal solution of the searching problems.

The effectiveness of the proposed algorithm has been experimented in this part.

Our proposed algorithm has been implemented using multi pascal language (Fig.5) and has been applied to a variety of graph coloring instances.

The GCP instances used in this section are from a benchmarking website formally named DIMACS graphs[1].

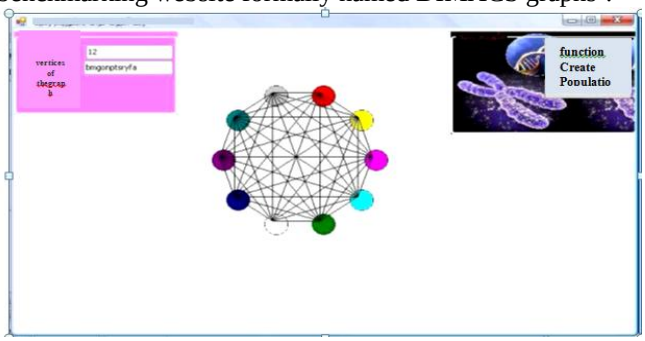

Fig. 5: Graph coloring by genetic coloring with 12vertices.

Next, we have compared our algorithm with the parallel genetic-tabu algorithm (PGTA) designed to solve GCPs (10). In terms of the resulting chromatic number, both algorithms return the same result, except for the problem instancequeen7 7.col, that the PGTA returns 7 while the HPGAGCP returns 8. Figure 6 shows the comparative results of our proposed algorithm and PGTA with 24 processors in terms of runtime. According to the figure, the results of our proposed algorithm are much better in all cases.

Using this monitoring tool, the user can watch how many CPUs are available, which of them are online, their average load, etc.

Figures 7 and 8 show the computational performance of various population sizes over various numbers of CPUs.

The population sizes are 5000, 10000individuals. Each data set was executed 3 times. The times reported are the average computation time, the average communication time and the average total time.

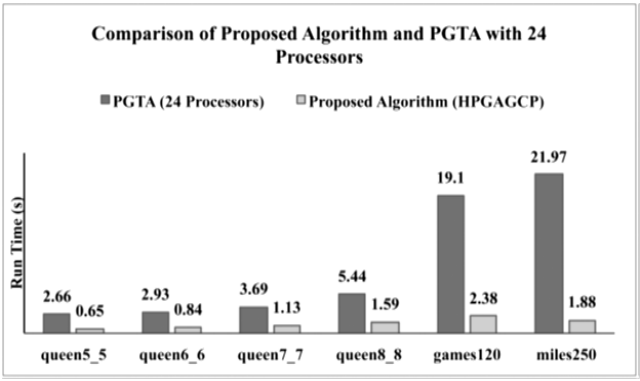

Fig.6: Comparison of the proposed algorithm and PGTA with 24 Processors.

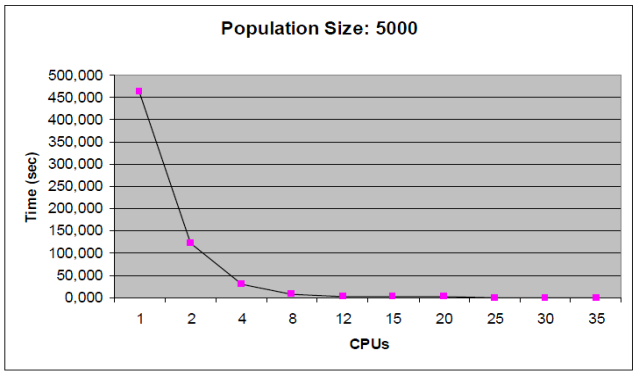

Fig.7: Average total time (in seconds) for various numbers of CPUs of population size 5000 individuals.

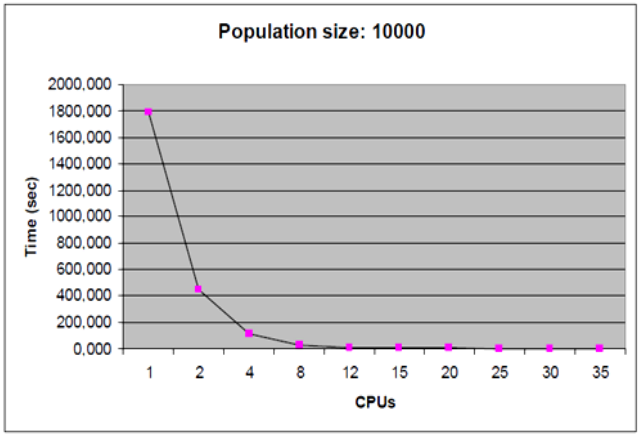

Fig. 8. Average total time (in seconds) for various numbers of CPUs of population size 10000 individuals.

Looking at Table 1, comparing total time for 1 CPU and total time for 35 CPUs, we can understand the big difference in time, and how many times faster the results

occur with parallelization. To be more specific with 35 CPUs and a population size of 5000 individuals, we can achieve a speedup of 512 times (total time cpu [1]/total time cpu(5).

Figure 2 also shows something similar. For the 2ndexperiment, the sample was doubled, from 5000 to 10000. Even the result with 35 CPUs here took 4 times more compared with the results of the 5000 sample. Time for 1 CPU was also increased, almost to 4 times.

Table 1.POPULATION SIZE 5000: TIME (IN SECONDS)

CPU	Computation	Time communication	Time total time
1	*463,370*	0,000	462,370
2	*121,748*	0,153	121,800
4	*29.205*	0,022	28,227
8	*7.386*	0,029	7,415
12	*3,455*	0,042	3,187
15	*2,541*	0,040	2,581
20	*1,468*	0,043	1,511
25	*1,139*	0,027	1,157
30	*0,958*	0,034	0,982
35	*0.868*	0,036	0,903

Another thing to admire is that when the number of CPUs decreases, times don't decrease in the same frequency. From 8 CPUs to 4, total time didn't increased by 2 times. From 28.50 seconds rose up to 111.43, almost 4 times bigger. So when the number of CPUs was divided to 2, total time, as well a computational time, was quadrupled. Something similar happened from 4 CPUs to 2 CPUs.

VII. CONCLUSION AND DISCUSSION AND FUTURE WORK

It seems that genetic algorithm can find an ideal answer for graph coloring while it's efficiency depends on the way of encoding crossover and mutation .

It seems that using matrix show and heuristic is better than the other method and closer to the actual answer. genetic algorithm is better than the other methods for graph coloring. we know that the best non genetic algorithm is presented for coloring of the special manners of genetic algorithm.

In this paper after producing a child its value is examined immediately and kept in a part of chromosome then after using some computation on the graph if the chromosome value is equal to graph the chromosome is chosen as the answer and the algorithm ends.

Simulation experiments reported in the paper provide an evidence that parallel genetic algorithms can be efficiently used for a class of graph coloring problems.

Additionally, genetic algorithms are proved to be problems where parallel techniques can take place, and our results assure the above statement. These remarkable results are the consequence of the proper fitness function, which in our situation seems to be good enough.

The importance of these results has to do with the data set. The algorithms described here can also be applied to the various subsets of the general GCP. High-performance computing still seems to be the discipline of computer science to find out solutions on such kind of problems. Using several real world data sets, scientists may come to important results. The genetic algorithm provides new hypo these is by continuous changing and mixing the pieces instead of searching general to specific or simple to complex hypo these is in each phase a collection of hypo these is which is called population is provided by replacing a part of present population with the children produced from the best hypo these is. we hope that by presenting more ideal methods for mutation crossover encoding then would be better solutions for graph coloring as well.

Using PSO algorithm is a method which can be used in the future. This method is evolutionary ins pined by the social behavior of the bird flocks or fish groups.

REFERENCES

1. A. Hertz , and D. de Werra, Using tabu search techniques for graph coloring, Computing, vol. 39, no. 4, pp. 345-351, 1987.
2. E. Burke and S. Petrovic, Recent research directions in automate timetabling, European Journal of operation research, vol. 140, no. 2,pp. 266-280, 2002.
3. Glass, C. A.|Prugel-Bennett, Genetic algorithm for graph coloring: Exploration of Galinier and Hao's algorithm, J. Combinatorial Optimization, pp,. 229-236, 2009.
4. D. Lim, Y-S. Ong, Y. Jin, B. Sendhoff and B-S. Lee, Efficient hierarchical parallel genetic algorithms using Grid computing, Future Generation Computer Systems, vol. 23, pp. 658-670, 2010.
5. Z. Konfrst, Parallel genetic algorithms: Advances, Computing trends, Applications and Perspectives, Proc. of the 18th International Parallel and Distributed Processing Symposium (IPDPS'04), 2011.
6. Ashby, Leif H., and Yampolskiy, Roman V. , Genetic Algorithm and Wisdom of Artificial Crowds Algorithm Applied to Light Up, The 16th International Conference on Computer Games: AI, Animation, Mobile, Interactive, Multimedia, Educational & Serious Games, Louisville, KY, USA: pp. 27-30, 2011.
7. J. Chen, E. Antipov, B. Lemieux, W. Cedeno, and D. H. Wood, DNA computing implementing genetic algorithms. In L. F. Landweber, E. Winfree, R. Lipton, and S. Freeland, editors, Evolution as Computation, pages 39--49, New York, Springer Verlag, 2008.
8. Burke, E., Newall, J, A Multi-stage Evolutionary Algorithm for the Timetable Problem. IEEE Trans. Evol. Comput, pp. 63–74, 2007.
9. Carter, M., Laporte, G., Lee, S.Y, Examination Timetabling: Algorithmic Strategies and Applications. J. Oper. Res. Soc. 47 , pp. 373–383, 2006.
10. B. B. Mabrouk, H. Hasni, and Z. Mahjoub, On aparallel genetic-tabu search based algorithm forsolving the graph colouring problem. European Journalof Operational Research, 197(3):1192–1201, 2009.