J. Basic. Appl. Sci. Res., 3(4)292-296, 2013

ISSN 2090-4304
Journal of Basic and Applied Scientific Research
www.textroad.com

Solving the Graph Coloring Problem by Modified Shuffled Frog Leaping Algorithm

Akhtar Hazratibishak[*1], Zahra Sadat Ghandriz[2], Jafar Sheykhzadeh[3], Sara Esfandiari[4]

[1,3]Department of Computer Engineering, Ahar Branch, Islamic Azad University, Ahar, Iran
[2]Department of Computer, Science and Research branch,Islamic Azad University, Yazd, Iran
[4]Department of Computer Engineering and Information Technology, Islamic Azad University, Qazvin Branch, Qazvin, Iran

ABSTRACT

A proper coloring of a graph is defined as a function of vertices of graph into a set of colors such that both adjacent vertices in the graph have different colors. Graph coloring problem is a NP-Complete issue; so in a short time many algorithms are generated that produce acceptable answers for solving it. In this paper we present a new method using a Modified Shuffled Frog Leaping Algorithm(MSFLA) for solve the graph coloring problem. The Proposed algorithms are compared with Caramia, Malaguti Fanabiki algorithms. Based on the results of sampled graphs, it is observed that the proposed algorithm produces much better solution in acceptable time.
KEYWORDS: Graph coloring problem, Modified Shuffled Frog Leaping Algorithm, Evolutionary algorithms.

1. INTRODUCTION

Coloring the vertices of graph problem is assign one color to each of the vertices of graph; such that any two adjacent vertices of the graph are assigned with different color. Graph coloring problem has many applications, including Timing issues, Register allocation problem, Solving partial differential equations, Parallel algorithms for Gauss – Saydl to solve the nonlinear algebraic equations, Extraction parallel sections of algorithms and Frequency assignment (Redl, 2004; Hale, 1980; Park et al, 1996; Maniezzo et al, 2000).Due to the importance of graph coloring problem and its many applications, many algorithms has been proposed for finding a valid coloring of the graph, including Accurate algorithms (Rozen et al, 2000; Welsh et al, 1967) Distributed algorithms (Awerbuch et al, 1989; Linial, 1992), Parallel algorithms (Jones et al, 1993;Luby, 1989), Approximation algorithms (Blum, 1991; Karger et al, 1998; Halperin et al, 2001) And heuristic algorithms (caramia et al, 1999; Funabiki et al, 1990; Malaguti et al, 2005). caramia and et al., (1999) have proposed an HCD algorithm that is a local search algorithm with priority rules that Takes advantage of Intelligence search techniques. This algorithm works in optimized version and the algorithm stop when a given number of iterations were obtained. Fanabiki and et al., (1990) presented a MIPS-CLR algorithm. That work in optimized version, but for graph coloring, requires one input value as the target (k_{init}). If the algorithm is not able to solve the problem with k_{init} color, then the input value is changed dynamically. Malaguti and et al., (2005) is presented the MMT algorithm that works based on a heuristic approach which has two phases: The first phase is based on an evolutionary algorithm while the second phase is a next phase of the optimization, based on the SCF.

Unlike traditional search algorithms, evolutionary computation techniques operate on a set of solutions in the search space and by creating cooperation and competition between solutions can quickly find optimal solution for complex optimization problems. These techniques mainly have been inspired by evolutionary process in nature that four famous cases of them is: Genetic Algorithms, Evolutionary Programming, evolutionary strategy and Genetic Programming. Except evolutionary computing techniques inspired from evolutionary process in natural, new computational techniques have been developed that simulate the social behavior like Ant Colony and Particle Swarm Optimization (PSO).

In this paper is introduced a new evolutionary algorithm to solve graph coloring problem based on Modified Shuffled Frog Leaping. The proposed algorithm has been inspired of the group of frog's life when looking for food. In this algorithm each frog represents a solution of the problem and initially the frog was divided into several groups, which based on this division in this method, there are two types of search techniques: The first technique that associated to the Frogs within each group is a local search technique that. The second technique related to competition between the groups for obtaining food (solution).

The remainder of the paper was organized as follows: In Section 2 and 3, has been discussed graph coloring problem and frog leaping algorithm respectively. In Section 4, proposed approach is presented. Results of the implementation are presented in section 5. final section is conclusions of paper.

***Corresponding Author:** Akhtar Hazratibishak, Department of Computer Engineering, Ahar Branch, Islamic Azad University, Ahar, Iran E-mail:a-hazrati@iau-ahar.ac.ir

2. Graph coloring

Assume G= (V, E), is a sample graph that in which V is a set of vertices and *E* [*V*] is a set of edges in graph G. Pure Vertex Coloring Problem (VCP) of the graph is mapping as a $C: V \rightarrow S$ Such that for two neighboring vertex V and W, C (v) ≠C (w). Members of the S consist of colors. The smallest integer value of k for G in the mapping C: V→ {1, 2... K} is called Chromatic number of G that is shown with χ (G). A graph G, is called k-chromatic if χ (G) =k; and if χ (G) ≤k then G is called k-colorable. Find the chromatic number of graph for k-colorable graphs, that k ≥ 3, is a NP-Complete problem (Karp, 1972).

Permissible coloring is to assign a color to the vertices of the graph; such that neighboring vertices have different color and optimum graph coloring equal to the minimum number of colors that needed for a permissible coloring. When assumptions are considered for graph coloring problem many different types of problem occur (Jensen et al, 1995). In this context, extensive research has been done which their results are the much broad number of exact and approximate, Heuristic and Metaheuristic algorithms (Kubale, 1998; De Werra, 1990).

3. Description of the modified shuffled frog leaping algorithm

3.1 SHUFFLED FROG LEAPING ALGORITHM (SFLA)

First time, Shuffled Frog Leaping Algorithm (SFLA) was expressed by Eusuff and Lansey in 2006(Eusuff et al, 2006).The main objective of expression the SFLA algorithm is to obtain an algorithm that can be solving the NP-Complete optimization problems, without using of mathematical equations. The Frog algorithm is a combination of GA based on memetic algorithm (MA) and Particle Swarm Optimization (PSO) algorithms. Indeed it can be said SFLA is produced of combining advantages of the MA and PSO algorithms. These algorithms was inspired from lives of a group of frogs when search for food. In this algorithm, every frog is a solution of the problem. The SFLA include initial population of possible solution. These solutions in fact are a collection of virtual frogs which are divided into several groups. Features of each group are affected by other groups. Also, like a particle swarm algorithm, SFLA has a local search; in each group frogs exchange information between each other, to improved its position with respect to the food. Then after any local search the obtained information of groups is compared together(Alinia Ahandani et al, 2009; Elbeltagi et al, 2007; Niknam et al, 2010).

The following steps are performed in the above method:

1. Initial population generate randomly. The initial population is consists of P frogs (P solutions).
   After generate the Initial population, this population must be arranged from best to worst solution based evaluation function
2. The frogs divided into m group, each group includes n frog such that $p = m \times n$.
   Way of division frogs into groups is that the first frog of sorted population assigned to the first group, the second frog assigned to the second group and m^{th} fog of this population is assigned to the m^{th} group. For m $+1^{th}$ frog in the sorted population again refer to first group and assign the m $+1^{th}$ frog to the first group and Similarly Continue until the n frog is placed in each of the m groups.
3. In this phase, local search is done; determine frog with the worst and the best solution in each group and show with $X_w$and X_b respectively. Also we determine the Frog with best solution in total population and show it's with X_g. Then location of worst frog (X_w) in each group, is modified As follows:

$$D_i = rand \times (X_b - X_w) \quad (1)$$

$$Xw(new) = Xw(old) + D_i \quad (2)$$

$$(-D\max \le D_i \le D\max)$$

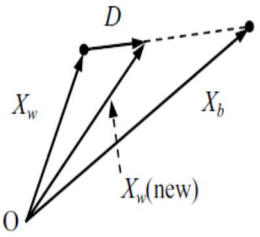

Fig 1.The basic rule Frog Leaping

Where rand is a random number between 0 and 1 and D_{max}represents the maximum displacement that one frog can have. If during this phases, frog in new location (X_w (new)) has a better response compared to his previous location (X_w (old)), then the previous location of the frog are replaced with the new location,And if the response does not improved then equations (1 and 2) are repeated, With the difference that in the new

state instead of X_b in equations we use of X_g. Even if with this change the response did not improve, we generate a new frog (solution) randomly and replaced it with previous Frog (X_w). Because all the elements of the vector X are integers, then each time after apply equations (9 and 10) should be rounded the obtained solution.

4. Step 3 is repeated for a number of iteration that predetermined.
5. At this stage after improvement location of frogs, new population sorted from best solution up to the worst solution.
6. If the stop condition of the algorithm is achieved, then stopped the algorithm, otherwise, return to the Step 2.

3.2MODIFIED SHUFFLED FROG LEAPING ALGORITHM

In frog leap algorithm that has been shown in Figure 1, when the frog with the worst fitness improve its position relative to the group or best frog, are placed along the line between X_b and X_w, which makes it possible that algorithm lead to wrong solution. For this reason, a method is introduced to improve this algorithm. The main idea of this method (Kubale, 1998) is a spreading direction and length of each frog's jump, so avoids the wrong convergent of algorithm.

In this method, if the first four frogs in the sorted frogs $\left(X_{g,4}, X_{g,3}, X_{g,2}, X_{g,1}\right)$, be selected randomly such that $X_{g,4} \neq X_{g,3} \neq X_{g,2} \neq X_{g,1}$, then the frogs' location vector change as follows:

$$X_{change} = X_{g,1} + rand_1 \times (X_{g,2} - X_{g,3}) + rand_2 \times (X_{g,1} - X_{g,4}) \quad (3)$$

$$X_{w,j}^{new} = \begin{cases} X_{change} \text{ if } rand_3 \leq rand_4 \text{ or } j = randperm \\ X_g \quad otherwise \end{cases} \quad (4)$$

Where: t represents the iteration number and randperm is a random number between 0 and 1 that indicates the number of categories. Also, rand1, rand2, rand3 and rand4 are random numbers between 0 and 1.
If these changes will lead to a better solution, it replaces with previous solution; otherwise it generates a random solution and replaces it with previous answer.

4. Use of modified shuffled frog leaping algorithm to solve the graph coloring problem

To solve the problem, at first we generate the initial population of frogs. Location of each frog is considered as a string of existing colors that it is assigned to graph vertices in pre-determined order. For example if location of a frog in a four-dimensional space to solve the problem of coloring a graph with five vertices is as <2, 1, 3, 2, 1> string, this string show that: the second and fifth graph vertices have first color in string and the first and four vertices have second color in string also third vertex has third color in string. So, in fact each frog isan N-dimensional vector, where N is the number of vertices of the graph.

After generate the initial population, the population must be sort based on fitness function. Fitness function, must be determining that assigning colors to vertices of the graph,whether or not satisfy the constraints.I.e. in assigning colors, each two adjacent vertex have different colors or not. This function returns the value of conflict, which reflects the number of vertexes that have similar colors. Since location of each frog equal to the assigned colors to the edges and according to the equations (1) and (2) it can be negative, so a function is needed to create relation between location and assigned colors. Always this function ensures that the assigned colors begin from number 1.
If the frog Location of population be X = (n_1, n_2… n_N) then fitness function for x is:

$$f(x) = \begin{cases} \max(n_i)_i^N & if\ conflict = 0 \\ conflict \times p + \max(n_i)_i^N & else \end{cases} \quad (5)$$

Which p is the degree of importance of the problem constraints that is not satisfies. Search space in graph coloring problem is a discrete space; therefore, components of each individual in population cannot be arbitrary value and must be limited to the natural numbers from 1 to N.

5. EXPERIMENTS AND RESULTS

Note that all experiments are done with the software Matlab 2010, Windows 7 Professional operating system with a system characterized by a Core i3, 2.13MHz RAM plus 4 Gigabyte. Experiments tested on the samples that

suggested in DIMACS (Johnson et al, 1996;ftp://dimacs.rutgers.edu/pub/challenge/graph/). The results of the proposed algorithms are compared with results of Caramia (1999), Fanabiki(1990) and Malaguti(2005) algorithms. Measure of evaluation for algorithms is the number of colors that is used for coloring. Any of the reported results in Table 1, is average of 25 times run. The first column (| V |) and second (| E |) in Table 1 are the number of vertices and edges of graphs, respectively. As can be seen in several of case, the proposed algorithm reduces the number of colors. In most cases (except two cases), the number of colors that obtained by the algorithm equal to the minimum number of colors that obtained by three algorithms were compared. When the number of vertices is increases, the population size of proposed algorithm increased from 300 to 700 individuals; Initial position of individual chosen randomly.

Table 1: The number of colors for coloring the graph using various algorithms

graph	DSJC125.5	DSJC500.9	DSJC1000.5	r125.1	r125.1c	Flat300-28-0	School1-nsh	le450-5a	le450-15d	flat1000-60-0
|V|	125	500	1000	125	125	300	352	450	450	1000
|E|	3891	112437	249826	209	7501	21695	14612	5714	16750	245830
Proposed algorithm	**9**	**17**	**72**	**5**	**47**	**31**	**14**	**5**	**16**	**58**
Caramia[14]	19	18	95	5	46	33	14	8	17	93
Fanabiki[15]	17	15	88	5	46	31	14	5	15	60
Malaguti[16]	5	15	84	5	46	31	14	5	15	60

6. Conclusion

For solve a graph coloring problem, in this article proposed new method using improved frog leaping algorithm. The proposed algorithm compared with other algorithms and results show that proposed algorithm produces better solution than the other methods. Proposed method works well in solve the graph with small size. Various experiments shown that the final answer is affected by the population size; such that when population size increased, algorithm can be achieved better optimization solution, however, its run rate comes down.

REFERENCES

1. Redl, T.A., (2004), "A Study of University Timetabling that Blends Graph Coloring with the Satisfaction of Various Essential and Preferential Conditions", PhD. Thesis, Rice University.
2. Hale, W.K., (1980), "Frequency Assignment: Theory and Applications", Proceeding of IEEE, Vol. 68, No. 12, pp. 1497-1514.
3. Park, T., and C.Y. Lee, (1996), "Application of the Graph Coloring Algorithm to the Frequency Assignment Problem", Journal of the Operations Research Society of Japan, Vol. 39, No. 2, pp. 258-265.
4. Maniezzo, V., and R. Montemanni, (2000), "An Exact Algorithm for the Min-Interference Frequency Assignment Problem", Tech. Report WP-CO0003, Scienzedell'Informazione, University of Bologna, Cesena, Italy.
5. Rozen, K.H., and et al, (2000), "Handbook of Discrete and Combinatorial Mathematics", United States of America, CRC Press.
6. Welsh, D.J.A., and M.B. Powell, (1967), "An Upper Bound for the Chromatic Number of a Graph and its Application to Timetabling Problems", Computer Journal, Vol. 10, pp. 85-86.
7. Awerbuch, B., and et al., (1989), "Network Decomposition and Locality in Distributed Computation", in Proceedings of the 30th Symposium on Foundations of Computer Science(FOCS), pp. 364–369.
8. Linial, N., (1992), "Locality in Distributed Graph Algorithms", SIAM Journal on Computing, Vol. 21, No. 1, pp. 193–201.
9. Jones, M.T., and P.E. Plassmann, (1993), "A Parallel Graph Coloring Heuristic", SIAM Journal of Scientific Computing, Vol. 4, p. 654.
10. Luby, M., (1989), "A Simple Parallel Algorithm for the Maximal Independent Set Problem", SIAM Journal on Computing, Vol. 4, p. 1036.
11. Blum, A., (1991), "Algorithms for Approximate Graph Coloring", PhD. Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.
12. Karger, D., R. Motwani, M. Sudan, (1998), "Approximate Graph Coloring by Semidefinite Programming", Journal of the Association for Computing Machinery, Vol. 45, No. 2, pp. 246-265.
13. Halperin, E., R. Nathaniel, U. Zwick, (2001), "Coloring k- Colorable Graphs using Smaller Palettes", In Proceedings of SODA, pp. 319-326.
14. Caramia, M., and P. Dell'Olmo, (1999),"A Fast and Simple Local Search for Graph Coloring", Proc. of the 3d Workshop on Algorithm Engineering WAE'99, Lecture Notes in Computer Science, pp. 319-313.

15. Funabiki, N., and T. Higashino, (1990), “A Minimal-State Processing Search Algorithm for Graph Coloring Problems”, IEICE Trans, Fundamentals, pp. 1420- 1430.
16. Malaguti, E., M. Monaci, and P. Toth, (2005), “A Metaheuristic Approach for the Vertex Coloring Problem”, Technical Report OR/05/3, DEIS University of Bologna.
17. Karp, R., (1972), "Reducibility among Combinatorial Problems", Complexity of computer computations,pp. 85-104.
18. Jensen, T.R., B. Toft, (1995), "Graph Coloring Problems", Wiley Interscience Series in Discrete Mathematics and Optimization.
19. Kubale, M., (1998), "Introduction to Computational Complexity and Algorithmic Graph Coloring", GdanskieTowarzystwoNaukowe.
20. De Werra, D., (1990), "Heuristics for Graph Coloring", Computing Suppl. 7, pp. 191-208.
21. Eusuff, M. M., K. Lansey, F. Pasha, (2006),” Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization,” Engineering Optimization, vol. 38, no. 2, pp.129–154.
22. AliniaAhandani, M., N. PourqorbanShirjoposht. R. Banimahd. (2009), “Job-Shop Scheduling Using Hybrid Shuffled Frog Leaping,” Proceedings of the 14th International CSI Computer Conference IEEE.
23. Elbeltagi, E., T. Hegazy, and D. Grierson, (2007). “A Modified Shuffled-Frog-Leaping Optimization Algorithm: Applications to Project Management.” Journal of Structure and Infrastructure Engineering, Taylor & Francis, 3(1), 53-60.
24. Niknam, T., E. AzadFarsani. (2010), “A hybrid self-adaptive particle swarm optimization and modified shuffled frog leaping algorithm for distribution feeder reconfiguration.” Engineering Applications of Artificial Intelligence, in press.
25. ftp://dimacs.rutgers.edu/pub/challenge/graph/.
    Johnson, D.S., and M.A. Trick (eds.), (1996), “Cliques, Coloring, and Satisfiability: 2nd DI-MACS Implementation Challange, 1993”, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society.