IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 8, August 2017

Using Shuffle Frog Leaping in Partitioning Graph

Somaye Amiri[1], Ali Hanani[2]

Computer Engineering, Islamic Azad University, Kermanshah, Iran[1,2]

Abstract: Graph partitioning is one of the most important problem in optimization and graph theory. This problem falls into NP-problems and does not have an exact solution. In this paper, we intend to propose a new solution to optimize balanced undirected graph partitioning problem. The solution is an evolutionary algorithm, so called Shuffle Frog Leaping. A fitness function will be used to evaluate the suitability of each frog in each iteration. In order to measure the performance of our method, we partitioned a random graph with 10000 nodes into arbitrary numbers. Furthermore, we compare our result with two well-known algorithms called Genetic algorithm and Artificial intelligent.

Keywords: Balanced Graph partitioning, Shuffle Frog Leaping, Optimization algorithms.

I. INTRODUCTION

The Graph partitioning [23] is one of the well-known problems in mathematics which has been used in many fields such as image processing, parallel processing [6, 7], parallel depth -first-search [9], complicated network, operating system, biological network [3, 4], geographically embedded networks [10], VLSI physical design, route planning [18, 20], air traffic control, social network [5]. GP has wide spectrum of types, but the general idea is breaking down a given graph into smaller pieces. The graphs can be directed, non-directed, weighted, or even mesh [9] and hyper graph. This paper addressed the k-balance partitioning problem for directed and unweighted graphs which has been defined in [32,]. Given a number $k \in N$ and $A = (V, E)$is a non-directed connected graph which $V = \{V_i, \ i = 1 \dots n\}$. GP asks for sub-graphs $(A_{1,} A_2, \dots, A_k\}$ that the following value is minimized:

$$\text{Min } \{|V1|, |V2|, \dots |Vk|\}$$

The problem cut a non-directed connected graph into a balanced blocks of vertices that total number of them in each block must be optimally equal. Due to the wide range of fields that using GP, it becames more and more crucial and challenging. Therefore, there are a lot of proposals have been presented in the literature to solve the problem including Exact Algorithms, Spectral partitioning [11, 12, 13, 14, 15, 30], Graph growing [16, 17], Flows [18, 20, 21], Tabu search [19, 23, 24, 25, 26], Geometric partitioning, Bubble framework and recursive [22], multilevel partitioning [24, 26, 27, 28], Evolutionary methods and further meta-heuristics like Genetic algorithms [20, 31, 32, 33].

In this paper we presented an efficient algorithm based on Shuffled Frog Leaping (SFL) optimization which is kind of evolutionary optimization algorithm. SFL has been proposed by Muzaffar Eusuf, et. al. [8] for addressing complex and large-scale combinatorial optimization problem. Due to its efficiency, researchers have been used it to achieve optimum or near-optimum solutions. In order to prove the performance of this algorithm, we compare the result with two other algorithms: Genetic Algorithm (GA), and Artificial Fish (AF).

Our paper is structured as follows. First, section two introduces shortly Shuffled Frog Leaping (SFL). In section 3, SFL is applied to the graph partitioning problem. Then section 4 represent results of performance checking of the proposed algorithm. Finally, points to future challenges.

II. SHUFFLED FROG LEAPING

The SFL inspired by natural memetics evolution which is based on population like most of the EAs. Set of virtual frogs interacting and divided into various memeplexes act as memes. A meme is a member of cultural evolution and different memeplexes are consider as different cultures of individual frogs. Each frog in memeplexes holds solutions which can effected by the idea of the other frogs and will be engaged by the process of mimetic evolution. The solutions spread among memeplexes which call it shuffling process. The process continue until the measurement condition are reached. The steps of the SFL algorithm are explained briefly bellow.

A. Global Search

First, number of memeplexes (m) and number of frogs (n) must be selected. Hence, the total size of frogs population is can be counted by $f = n \times m$. Then the frogs will be partitioned by this way: the first frog assigns to the first memeplex, the second one assigns to the second group, finally the m th frog goes to memeplex m. Then $m + 1$ assigns to the first group and so on.

B. Local search
After assigning frogs to the memeplexes, the best and worst frog by the following formula will be calculated:

$$D(i) = rand(x) \times \; (X_{best} - X_{worst})$$

The above formula computes the i th frog's position based on the difference between the best frog (X_{best})and the worst (X_{worst}) in memplex i . D(i) is used to improve the position of the worst frog by computing a new position. The following formula calculates a new position:

$$X_{newPositin} \; = \text{current position } X_w + D_i$$

$$D_{max} \le D_i \le -D_{max}$$

If the new position for the worst frog has been improved, then it will be exchanged by that. Otherwise, the calculation of new position will be repeated until the stop condition reached. At the end, if no better position found, a new frog will be generated randomly and exchanged with the current worst frog.

After repeating the above steps for few generation, all memeplexes are mixed together and sort based on their fitness and re-assign like in the global search section.

III.PROPOSED METHOD

In this section, we present using shuffle frog leaping to partition graphs. Each frog is representing an one dimension border includes permutation of graph's edges and the length of each frog is equal to the number of graph's edges. In the first step, we generate a random population of frogs which the size of population is f. The following frog is a random frog that is generated from the Figure 1.
$frog_1 = [5, 3, 11, 4, 9, 2, 8, 6, 1, 12, 10, 7]$

For partition the graph in Figure 1. Into for example 4 parts ($k = 4$), we generate $f = 9$ frogs and assign them into three groups $m = 3$ memplex which each memplex has $n = 3$ frogs.

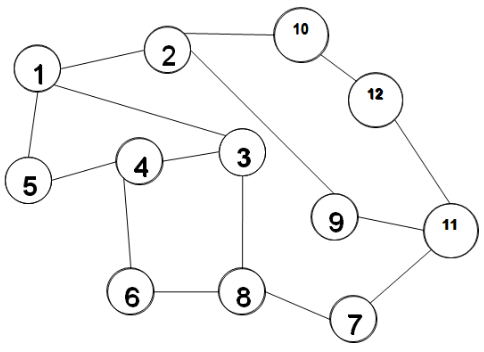

Figure 1 A random graph with 12 nodes.

To partition a single frog the following steps should be done:
1- selecting the leaders of each part. In order to partition the graph, we select some leaders. The number of leaders is $k = 3$. It should be mentioned that the users choose the amount of k.
2- starting to assign the $(k + 1)^{th}$ node of the frog to the leaders.
3- the nodes must have a connection to the leaders in the original graph.
4- if a node has connection to more than one leader, then we assign it based on the priority or based on the amount of nodes that the leaders have.
5- if a node does not have any connection to the leaders, it remains unchanged until the by one another node indirectly be can assigned.
6- these steps will be continued until all nodes have been assigned.

At the end, the fitness must be calculated. The following formula is the fitness function:

$$\text{Fitness Function } = \frac{K \times T}{|V|}$$

Where k is the number of parts, T is the number of edges within the biggest part, and v is the number of edges that the original graph has. The best frog and the worst frog are indicated by the highest and the lowest amount of the fitness function. Following table shows the Pseudo code of the proposed method.

```
Shuffle frog based partitioning
Generating random frogs by permutation of the given graph nodes;
While the stop condition reached
{
Assign the frogs into m memplexes;
Partition $frog_0$ to $frog_n$;
Calculate frogs Fitness functions;
Sort frogs based on their fatnesses;
Generate a new position for the worst frog in all memplexes;
Mix all memplexes, sort frogs again;
Write the best frog into the blackboard;
Reassign them into memplexes;
}
Return the best frog.
```

IV. EXPERIMENT AND RESULT

In order to measure the performance of the propose method, we generate a random graph with 10000 nodes and tried to partition it into 100 and 500 parts. As well as, we compare the results with two other well-known evolutionary algorithms so called Genetic and Artificial fish.

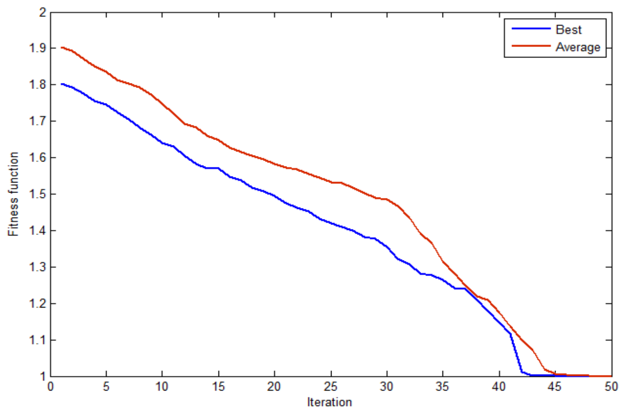

Figure 2 Convergence history of mean and best for partitioning into 100 parts.

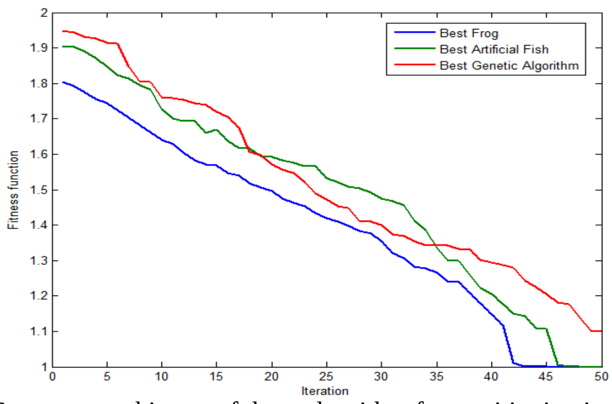

Figure 3 Convergence history of three algorithm for partitioning into 100 parts.

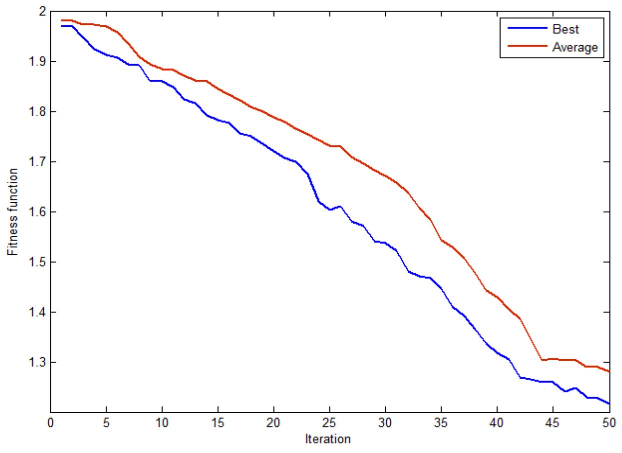

Figure 3 Convergence history of mean and best for partitioning into 500 parts.

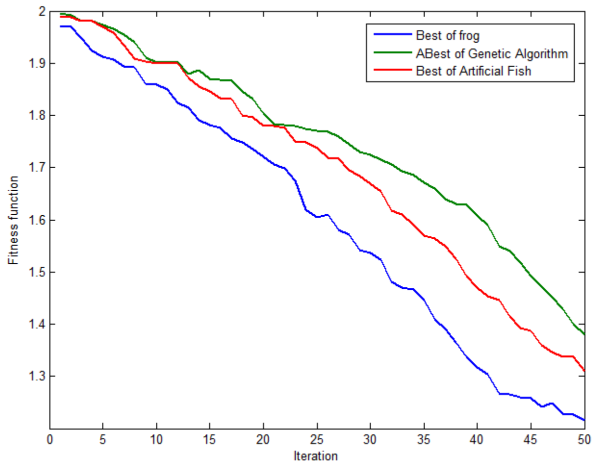

Figure 4 Convergence history of three algorithm for partitioning into 500 parts.

The best result of 50 times run has been summarized in the following table.

Number of partition	Algorithms		
	Shuffle Frog Leaping	Genetic algorithms	Artificial intelligent
100	1.12	1.60	1.48
500	1.22	1.75	1.62

V. CONCLUSION

The graph partitioning is one of the most important problems which has been appears in many fields. In this research, we have been used a new evolutionary algorithm so called Shuffle Frog Leaping (SFL) to balance partition of non-directed graphs. We present the problem in the way that can be solved with SFL and compute the fitness of frogs by a well-defined fitness function. In order to measure the performance of the proposed method, a random graph with 10000 nodes partitioned into arbitrary parts and the results compared by two well-known algorithm Genetic and Artificial fish. The results indicated that SFL is more optimum than others.

REFERENCES

[1] Amiri. S, Hanani. A, "Areview of graph partitioning applications", The first national conference of computer and electronic and computer, Azad university, Majlesi, Isfahan, Iran.
[2] Amiri. S, Hanani. A, "A systematic review on graph partitioning optimization", Azad univerisity, Sousan gerd, Khoozestan, Iran.
[3] B. Junker and F. Schreiber, (2008), “Analysis of Biological Networks”, Wiley.
[4] R. Mondaini. Biomat, (2009), “International Symposium on Mathematical and Computational Biology”, Brasilia, Brazil.
[5] S. Fortunato. (2009), “Community Detection in Graphs”, CoRR, abs/0906.0612.
[6] P. Erd˝os, A. Gy´arf´as, and L. Pyber, (1991), “Vertex coverings by monochromatic cycles and trees”, J. Combin. Theory, Ser. B 51, 90-95.

[7] C. Aykanat, B. B. Cambazoglu, F. Findik, and T. Kurc. (2007) “Adaptive Decomposition and Remapping Algorithms for Object-space-parallel Direct Volume Rendering of Unstructured Grids”, J. Parallel Distrib. Comput., 67(1):77_99.

[8] MUZAFFAR EUSUFF, KEVIN LANSEY, and FAYZUL PASHA, (2006) "Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization", Engineering Optimization Vol. 38, No. 2, , 129–154.

[9] M. Zhou, O. Sahni, et al. (2010), “Controlling Unstructured Mesh Partitions for Massively Parallel Simulations. SIAM Journal on Scienti_c Computing”, 32(6):3201_3227.

[10] U. Lauther. (2004), “An Extremely Fast, Exact Algorithm for Finding Shortest Paths in Static Networks with Geographical Background”, In Münster GI-Days.

[11] P. Sanders and C. Schulz, Think Locally, (2013), “Act Globally: Highly Balanced Graph Partitioning”, SEA.

[12] S. T. Barnard and H. D. Simon. (1993), “A Fast Multilevel Implementation of Recursive Spectral Bisection for Partitioning Unstructured Problems”, In 6th SIAM Conference on Parallel Processing for Scienti_c Computing, pages 711_718.

[13] C. Lanczos. (1950), “An Iteration Method for the Solution of the Eigenvalue Problem of Linear Di_erential and Integral Operators. Journal of Research of the National Bureau of Standards”, 45(4):255_282.

[14] G. Karypis and V. Kumar. (1998), “A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs”, SIAM Journal on Scienti_c Computing, 20(1):359_392.

[15] A. George and J. W. H. Liu.(1981), “Computer Solution of Large Sparse Positive”, De_nite Systems. Prentice-Hall.

[16] L. R. Ford and D. R. Fulkerson. (1956), “Maximal Flow through a Network. Canadian Journal of Mathematics”, 8(3):399_404.

[17] C. Farhat and M. Lesoinne.(1993), “Automatic Partitioning of Unstructured Meshes for the Parallel Solution of Problems in Computational Mechanics”, Journal for Numerical Methods in Engineering, 36(5):745_764.

[18] R. D. Williams. (1991), “Performance of Dynamic Load Balancing Algorithms for Unstructured Mesh Calculations”, Concurrency: Practice and experience, 3(5):457_481.

[19] John R. Gilbert, Gray L. Miler, Sahng-Hua Teng, (1994), "Geometric Mesh Partitioning: Implemetation and Exprements", 1994.

[20] A. H. Land and A. G. Doig. (1960), “An Automatic Method of Solving Discrete Programming Problems. Econometrica”, 28(3):497_520.

[21] M. Fiduccia and R. M. Mattheyses. (1982), “A Linear-Time Heuristic for Improving Network Partitions”, In 19th Conference on Design Automation, pages 175_181.

[22] H. D. Simon and S. H. Teng. (1997), “How Good is Recursive Bisection? SIAM”, Journal on Scienti_c Computing, 18(5):1436_1445.

[23] Aidin B, Henning M, Ilya s, Peter S, and Cristian S, (2013), "Recent Advances in Graph Partitining".

[24] G. Karypis and V. Kumar. (1999), “Multilevel k-Way Hypergraph Partitioning”, In 36th ACM/IEEE Design Automation Conference, pages 343_348. ACM.

[25] Ü. Çatalyürek and C. Aykanat. (2011), “PaToH: Partitioning Tool for Hypergraphs”.

[26] G. Karypis and V. Kumar. (1996), “Parallel Multilevel k-way Partitioning Scheme for Irregular Graphs”, In ACM/IEEE Supercomputing'96.

[27] F. Pellegrini. Scotch Home Page. http://www.labri.fr/pelegrin/scotch.

[28] C. Chevalier and F. (2008), “Pellegrini. PT-Scotch: A Tool for E_cient Parallel Graph Ordering”, Parallel Computing, 34(6):318_331.

[29] B. Monien and S. Schamberger. (2004), “Graph Partitioning with the Party Library: Helpful-Sets in Practice”, In 16th Symposium on Computer Architecture and High Performance Computing, pages 198_205.

[30] Pavla Kabel´ıkov´, (2006), "Graph Partitioning Using Spectral Methods", Technical University of Ostrava.

[31] R. Rajabioun, (2011), “Cuckoo Optimization Algorithm” ; Elsevier; Applied Soft Computing, 5508–5518; 2011.

[32] Sina Zangbari Kouhi, Faranak Nejati, Jaber Karimpour, (2013), "Solving the Graph Partitioning based on Cuckoo Optimization Algorithm (COA)", International Journal of Advanced Research in Compute.

[33] A. Cincotti, V. Cutello, M. Pavone, (2001), "Graph Partitioning using Genetic Algorithms with ODPX ", Dept. of Mathematics and Computer Science University of Catania V.le A. Doria 6, 95125 Catania, Italy.