Vol 7. No. 1 - March, 2014
African Journal of Computing & ICT

www.ajocict.net

A Shuffled Frog-Leaping Algorithm for Optimal Software Project Planning

***R. O. Oladele & H. A. Mojeed**
Department of Computer Science
University of Ilorin
Ilorin, Kwara State, Nigeria
*roladele@yahoo.com, leye4k@yahoo.com

*_Corresponding author_

ABSTRACT
In recent time, software project management has received considerable attention from researchers in the field of Search Based Software Engineering (SBSE). This paper presents an approach to Search Based Software Project Planning based on Shuffled Frog-Leaping Algorithm (SFLA). Our approach seeks to optimize work package scheduling with a view to achieving early overall completion time. To evaluate the algorithm, it is tested on a set of randomly generated problems and it's results are compared with those of Genetic Algorithm (GA). Results indicate that SFLA is significantly superior to GA.

Keywords: Shuffled Frog-Leaping Algorithm, work package, software project planning, Search Based Software Engineering, Design structure matrix

African Journal of Computing & ICT Reference Format:
R. O. Oladele & H.A. Mojeed (2014): A Shuffled Frog-Leaping Algorithm for Optimal Software Project Planning.
Afr J. of Comp & ICTs. Vol 7, No. 1. Pp 147-152 .

1. INTRODUCTION

Software engineering projects cannot be completed on schedule and within budget unless good software project management techniques are enforced. However, a thorough planning of the progress of a project is crucial for effective management of the project. Planning a large scale software (or another type of) project involves Work Packages (WPs). A work package defines not just the work product but also the staffing requirements, duration, resources, name of the responsible individual, and acceptance criteria for the work product [2]. The work packages are usually obtained from a Work Breakdown Structure. Given a fixed number of WPs for a fixed number of projects, there exists an optimal WPs assignment to time-slots such that the project completion time is minimized. WP ordering, one can find an optimal staff distribution into teams. This is an NP-hard problem problem for which heuristic methods have proved to be effective and popular among other methods.

Barreto et al [3] applied constraint satisfaction to staff software projects. However, their goal differs from ours in that they aimed at allocating maintenance requests to the most qualified team in terms of skills, to the cheapest team, or to the team having the highest productivity. Bertolino et al[4] employed performance engineering technique, based on the use of queuing models and UML performance profiles, to assist project managers in making decisions related to organization of teams and tasks.

Karova et al [5] presented implementation of GA for Project Planning and Project Scheduling Problem. Their algorithm was tested on a set of randomly generated problems and their results show that GA can be used by project manager to better simulate realistic situations and reorder the WPs and delay the project deadline, if the need arises.

SFLA is a memetic meta-heuristic that is based on evolution of memes carried by interactive individuals and a global exchange of information among the frog population [6]. It combines the strengths of Memetic Algorithm (MA) and the social behaviour-based Particle Swarm Optimization (PSO) Algorithm. In SFLA, the population consists of frogs (solutions) that is partitioned into subsets referred to as memeplexes. The different memeplexes are considered different cultures of frogs, each performing a local search [7]. Within each memeplex, the individual frogs hold ideas, that can be influenced by the ideas of other frogs, and evolve through a process of memetic evolution. After a defined number of memetic evolution steps, ideas are passed among memeplexes in a shuffling process [8]. The local search and the shuffling processes are repeated until a specified convergence criterion is satisfied.

In this paper, we implemented SFLA and tested it on a set of randomly generated problems of software project planning. We make two primary contributions in this paper: (1) SFLA is applied to solve Software Project Planning Problem, and to the best of our knowledge, this is the first paper in the SBSE literature to employ SFLA. (2) The results of the application

of SFLA in comparison with GA are reported. The obtained results provide evidence to support the claim that SFLA is superior to GA

2. PROBLEM STATEMENT

Planning a large scale software project involves a set of activities called Work Packages (WPs) , and an allocation of programmers to teams and teams to work packages [9]. Given a fixed number of Work Packages, there exists an optimal WP ordering and optimal people distribution into teams. Such resource allocation problems are instances of the 'bin packing problem', which belong to the class of Non-deterministic Polynomial-time hard (NP-hard) problems. In this paper, we focus on finding an optimal WP scheduling: Given a project that consists of a set WP's ={wp_1, wp_2,……,wp_n} of tasks to be performed, a set of dependency constraints DEPS = {(wp_i,wp_j),…..(wp_{in},wp_{jn})}, such that 0 ≤i ≤ n and 0 ≤j ≤ n and j ≠i of dependencies between tasks, where wp_j requires wp_i to be completed first. We seek an optimal ordering of tasks in the sequence in which they should be completed without violating dependency constraints such that the overall project completion time is minimized.

3. PROBLEM MODELLING

The problem is modelled with a Design Matrix Structure (DSM), an efficient method which shows the relationships between the activities in a project. It can be represented as an *n* x *n* multi-dimensional array representing tasks and precedence rule. The diagonal elements represent the tasks and off diagonal elements specify the precedence relationships. Using a scheduling problem consisting of two software projects, each containing 5 WPs which represent the tasks involved in the development of the projects, the corresponding DSM can be represented as shown in Figure 1

Project 1

1				
	2			
	1	3		
			4	
1		1		5

project 2

6				
	7			
1		8		
1			9	
		1		10

Figure 1: DSM model of project scheduling problem.

From above figure, the DSM indicates that WP1 precedes WP5, WP2 precedes WP3, WP3 precedes WP5, WP6 precedes WP8 and WP9, and WP8 precedes WP10. The WPs are ordered according to the precedence rule modeled by the above DSM. For example the orderings below shows correct (a) and incorrect (b) schedules:

1	6	2	8	3	5	9	4	7	10

(a)

1	3	7	5	2	8	9	4	6	10

(b)

Figure 2: Correct and incorrect WP orderings.

4. SFLA DESIGN

In general, SFLA works as follows; First, an initial population of F frogs is created randomly. Afterwards, the frogs are sorted in a descending order according to their fitness. Then, the entire population is divided to form memeplexes, within each memeplex the frogs with the best and the worst fitness are identified as X_b and X_w respectively. Also, the frog with the global best fitness is identified as Xg. Then, a process is applied to improve only the frog with the worst fitness (not all frogs) in each cycle.

Accordingly, the position of the frog with the worst fitness is adjusted as follows [10]:

Change in frog position:

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

New position:

$$X_{new} = X_w + D_i \quad ; \quad -D_{max} \leq D_i \leq D_{max} \quad (2)$$

rand is a random number between 0 and 1, and D_{max} is the maximum allowed change in a frog's position. If this process produces a better solution, it is replaced for the worst frog. Otherwise, the calculations in equations (1) and (2) are repeated but with respect to the global best frog (i.e. X_b is replaced with X_g). If no improvement is possible, then a new solution is randomly generated to replace the worst frog. Hence, the calculations continue for a specific number of iterations [8]. The main parameters of SFLA are: population size F; number of memeplexes m; and number of shuffling iterations in each memeplex q.

4.1 SFLA For Optimal Project Planning

The SFLA approach in solving project planning and scheduling problems combines the local search within each memeplex and global information interchange from parallel local searches among all memplexes to move towards a global solution using population-based model of frogs which represents feasible solutions (correct WP orderings).

4.2 Individual Frog Representation

The position vector of each individual frog represents a feasible solution of WPs schedules. Each frog is encoded as an *n*-sized array; the value of each meme (element of the array) indicates the position of the WP in the incoming ordered sequence and the index value represents the WP itself. The population is a set of *F* frogs (*F* WP ordered lists). The frog schema is shown in figure 3.1

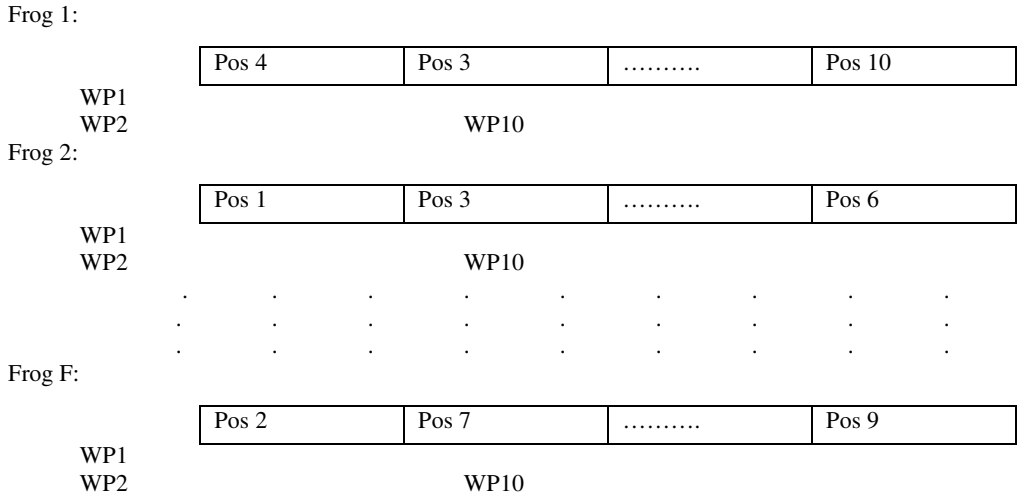

Figure 3: The frog schema.

4.3 Fitness Function

The fitness f of a frog is based on the constraints penalties. It is calculated as the sum of penalty points present in each frog with respect to the precedence rule predefined. Using the above DSM structure and the frog schema, the fitness values of two frogs A and B is given below:

1	3	5	8	6	2	9	4	7	10

wp1 wp2 wp3 wp4 wp5 wp6 wp7 wp8 wp9 wp10

Frog A

1	5	2	8	4	9	3	6	7	10

wp1 wp2 wp3 wp4 wp5 wp6 wp7 wp8 wp9 wp10

Frog B

f(A) = 0 (no violation of precedence rule)
f(B) = 3 (three violations of precedence rule)
The lower the value of f the fitter the frog since the fitness is calculated based on penalties.

4.4 Formation Of Memeplex

Memeplexes are constructed by partitioning the initial population of frogs. The entire population is divided into m memeplexes, each containing n frogs. In this process, the first frog goes to the first memeplex, the second frog goes to the second memeplex, frog m goes to the mth memeplex, and frog m+1 goes back to the first memeplex and so on as shown in Figure 4.

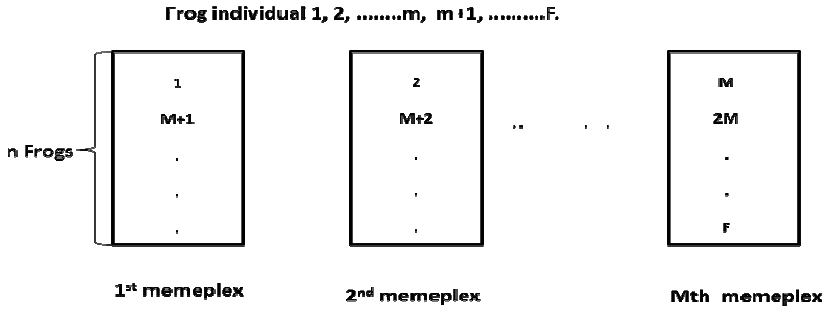

Figure 4: Formation of Memeplexes

4.5 Local Exploration

This is the part of the algorithm where the frog with worst performance in each memeplex is improved and updated. Within each memeplex, the worst performance frog is updated according to the following simple rule:

The new frog X_{new} is obtained by randomly selecting a subsequence in X_b to replace the corresponding position in X_w, while keeping the other positions in X_w unchanged or if violating the precedence constraints , just randomly relocate the remaining positions to form a new feasible solution. The idea is illustrated in Figure 5. If the fitness of X_{new} is better than that of X_w, then replace X_w with X_{new}, otherwise replace X_b with the global best X_g and carry out the same operation as the above to generate another new feasible solution X_{new} . If its fitness is better than that of X_w, then replace X_w with this new X_{new}, otherwise randomly generate a new feasible solution to replace X_w, where;

X_{new} = new updated frog, X_w = worst frog, X_b = best frog and X_g= global best frog

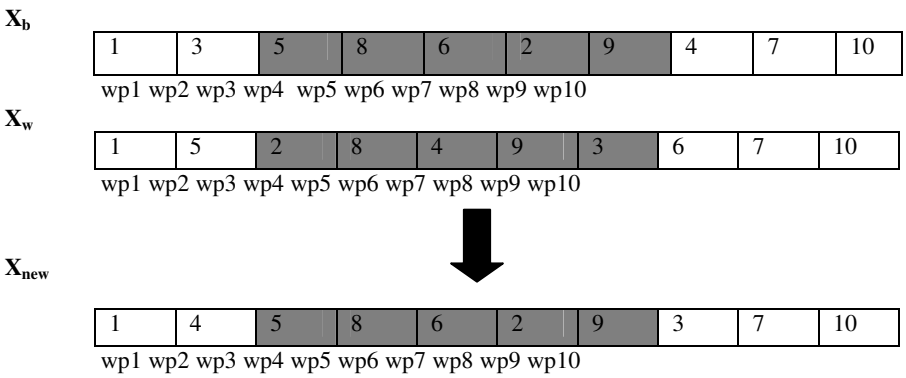

Figure 5: Update (X_{new}) in a Submemeplex

This operation is performed for a specific number of iteration. The number of iteration q here determines the time spent for local meme transference and in turn the efficiency of the local search. Intuitively we chose *q* to be dependent on the problem size with the value q = 2*n*, where n is the number of frogs in a memeplex.

4.6 Convergence

The convergence criterion we used is given by the formula:
$[|f(1)| - |f(F)|] < Є$

Where f represents the fitness and Є is the convergence tolerance**.** In order to ensure high convergence rate the value of Є is set to 1. The solution converges when at least 90 % of the frogs in the population have a fitness value 0.

5. COMPUTATIONAL EXPERIMENTS, RESULTS AND DISCUSSION

The algorithm is coded in JAVA and executed on HP 655 pavilion laptop with Windows 7 operating system, AMD E2-1800 APU 1.7 GHz CPU and 4.00GB RAM.
The proposed SFLA is tested on randomly generated problems with 10, 20, 30, and 50 WPs respectively. The number of memeplex is set to 5 and number of iterations per memeplex is 2n where n is the number of frogs in a memeplex. To avoid any misinterpretation of the optimization results related to the choice of any particular initial parameters, all results are obtained by averaging over 20 independent runs. The fitness value is given by the sum of penalty points and in each test the population size is varied as (30, 50, 80 and 100). The SFLA results and GA results extracted from [5] are presented in Table 1.

From the table of results, it is clear that SFLA is superior to GA. For GA, the best results are obtained when WP = 30. However, for SFLA the best results are obtained when WP =50, in fact, the larger the size of WP the better the result. The implication of these results is that, apart from the fact that SFLA is more effective than GA, it has an added advantage of being able to handle projects with larger size of WPs. For both SFLA and GA, the optimal value (optimal schedule) is obtained when the population size is 80.

Table 1: SFLA results versus GA results

Population size N	WP's	Fitness SFLA	Fitness GA [1]
30	10	14.80	16.1
	20	8.30	10.00
	30	3.35	3.50
	50	2.85	4.00
50	10	13.10	15.75
	20	7.30	9.80
	30	2.65	2.80
	50	2.35	3.40
80	10	12.15	15.55
	20	7.23	9.85
	30	2.45	2.60
	50	2.30	3.05
100	10	12.95	15.90
	20	8.22	9.60
	30	3.03	3.05
	50	2.75	4.25

6. CONCLUSION

The problem of assigning optimal WP to timeslots with a view to minimizing project completion time has been solved using SFLA. Experimental results show that the proposed SFLA is effective in finding optimal solution. Comparison of SFLA results with GA results reveals that SFLA outperforms GA. Results also show that while the best performance of GA occurs when WP = 30, the performance of SFLA improves as WP increases. In the future, the application of SFLA to multi-objective version of the problem will be considered as an extension of this study

REFERENCES

[1] Stylianou, C., Andreou, A. S., "*A multi-objective genetic algorithm for intelligent software project scheduling and team staffing*". Intelligent Decision Technologies 7(1): pp. 59 -80, 2013.

[2] Schach, S. R., "Object-Oriented and Classical Software Engineering", McGrawHill, 2002.

[3] Barreto, A., Barros, M. and Werner, L. "Staffing a Software Project: A Constraint Satisfaction and Optimization-based Approach.Computers and Operations Research, 2008 .

[4] Bertolino, A., Marchetti, E. and Mirandola, R. "Performance Measures for Supporting Project Manager Decisions. Software Process: Improvement and Practice, 12(2): pp. 141–164, 2007.

[5] Karova, M., Petkova, J., Smarkov, V. "*A Genetic Algorithm for Project Planning Problem"*, in Proceedings International Scientific Conference Computer Science'2008, pp. 647 – 651. 2008.

[6] Xue-hui L., Ye Y., Xia L. "*Solving TSP with Shuffled Frog-Leaping Algorithm"*. ISDA, vol. 3, pp.228-232, 2008. Eighth International Conference on Intelligent Systems Design and Applications, 2008

[7] Elbeltagi, E., Hegazy, T. and Grierson, D., *"Comparison among five evolutionary-based optimization algorithms"*. J. Adv. Engng. Informatics, 2005, 19, pp. 43 – 53

[8] Shepperd M.J. and Schofield C. "*Estimating software project effort using analogies*". IEEE Transactions on Software Engineering 23(11): pp. 736–743, 1997.

[9] Jalote, P., "*Software project management in practice",* Addison Wesley, 2004

[10] Eusuff, M.M. and Lansey, K.E., "*Optimization of water distribution network design using the shuffled frog leaping algorithm"*. J. Water Resour. Planning Mgmt, 2003, 129, pp. 210 - 225

[11] Gueorguiev S., Harman M., Antoniol G., "*Software Project Planning for Robustness and Completion Time in the Presence of Uncertainty using Multi-Objective Search Based Software Engineering*". in Proceedings of the annual conference on Genetic and evolutionary computation (GECCO'09), July, 8–12, 2009, Montréal Québec, Canada. 2009.

[12] Ren, J., Harman, M., and Penta, M. D., "*Cooperative Co-evolutionary Optimization of Software Project Staff Assignments and Job Scheduling*". M. B.Cohen and M. O. Cinneide (Eds.): SSBSE 2011, LNCS 6956, pp. 127-141, 2011.

[13] Maia, C. L. B., Nascimento, T. F., Freitas, F. G., Souza, J. T., "*An Evolutionary Optimization Approach to Software Test Case Allocation*", CIIT 2011, CCIS 250, pp.637-641, Springer-Verlag

[14] Ren, J., Harman, M., Penta, M. D., "*Cooperative co-evolutionary optimization of software project staff assignments and job scheduling*", SSBSE 2011, LNCS 6956, pp.127-141, Springer-Verlag

[15] Rodriguez, D., Ruiz, M., Riquelme, J. C., "*Multiobjective simulation of optimization in software project management*", in Proceedings of the 13th annual conference on Genetic and evolutionary computation (GECCO' 11), pp. 1883 - 1890, 2011.

APPENDIX: PSEUDOCODE FOR SFLA

```
Begin;
  Generate random population of P frogs;
  For each individual i in P: calculate fitness (i);
  Sort the population P in descending order of their fitness;
    Divide P into m memeplexes;
    For each memeplex;
      Determine the best and worst frogs;
      Improve the worst frog position
      Repeat for a specific number of iterations;
    End;
    Combine the evolved memeplexes;
    Sort the population P in descending order of their
fitness;
  Check if termination criterion is satisfied;
End;
```

Authors' Brief

Dr. R. O. Oladele is a Faculty at the Department of Computer Science, University of Ilorin, Nigeria. He obtained a Bachelor of Science Degree in Mathematics at the University of Ilorin, Ilorin, Nigeria in 1992, a Master of Science Degree in Mathematics at the University of Ilorin, Ilorin in 1998 and a PhD Degree in Computer Science from the University of Ilorin, Ilorin, Nigeria in 2013. His research interests are Algorithms, Combinatorics, Optimization, Computational Complexity Theory, and Software Engineering.

Mr. H. A. Mojeed is currently on national youth service at Kebbi State, Nigeria. He obtained a Bachelor of Science Degree in Computer Science at the University of Ilorin, Ilorin, Nigeria in 2013. His research interests are Algorithms, Software Engineering and Optimization.