An Investigation of Optimal Project Scheduling and Team Staffing in Software Development using Particle Swarm Optimization

Simos Gerasimou[1], Constantinos Stylianou[2] and Andreas S. Andreou[1]

[1]*Department of Computer Engineering and Informatics, Cyprus University of Technology, 31 Archbishop Kyprianou Ave., P.O. Box 50329, Lemesos, 3036, Cyprus*

[2]*Department of Computer Science, University of Cyprus, 75 Kallipoleos Ave., P.O. Box 20537, Lefkosia, 1678, Cyprus*

Keywords: Software Project Management, Project Scheduling, Team Staffing, Particle Swarm Optimization.

Abstract: Software development organizations often struggle to deliver projects on time, within budget and with the required quality. One possible cause of this problem is poor software project management and, in particular, inadequate project scheduling and ineffective team staffing. This paper investigates the application of a particle swarm optimization algorithm to help software project managers perform these activities effectively. Specifically, the proposed approach aims to create optimal project schedules by specifying the best sequence for executing a project's tasks and minimizing the total project duration. Simultaneously, it seeks to form skilful and productive working teams with the best utilization of developer skills. These considerations have been suitably encoded into the algorithm, with several hard constraints and objective functions appropriately formulated so as to assess the generated solutions with respect to their feasibility and also their quality. The initial results obtained are quite encouraging for the majority of the performed tests and indicate that the proposed approach is able to deal with the issues of scheduling and staffing in software project management.

1 INTRODUCTION

One of the serious problems concerning the majority of software development organisations is the high rate of software project failures. According to the Standish Group's CHAOS Report of 2009, only 32% of projects produced software systems that were delivered successfully on time and within budget and also provided the required features and functionality (Standish Group, 2009). These figures, give strong indications that software development companies systematically fail to accurately plan and properly measure their development processes, and the reasons leading to low success rates have, therefore, been the focal point of many software engineering researchers.

Among the most significant causes attributed to software project failures has been the insufficient and inappropriate practices followed by software project managers regarding project scheduling and team staffing activities. In the former case, incorrect estimates both before and during software development have been found to play a crucial role in software project delays and overruns, whereas in the latter case, assigning project tasks to less suitable project team members is one of the main causes of low quality end-products.

The research presented in this paper is an initial investigation to deal with these issues of software project management through a swarm intelligence approach that facilitates both the scheduling of project tasks and the allocation of the most suitable team members to tasks in an automated way. Specifically, the approach targets two goals. Firstly, to construct an optimal sequence of task executions and to help minimize software project duration without any violation of possible dependencies existing between tasks. Secondly, to form an efficient and operational software project team with the best possible utilization of skills measured in terms of developer experience.

2 RELATED WORK

There have been a number of approaches proposed

DOI: 10.5220/0004001001680171
In *Proceedings of the 14th International Conference on Enterprise Information Systems* (ICEIS-2012), pages 168-171
ISBN: 978-989-8565-11-2

over the years that aim at helping software project managers decide on various technical factors such as project duration and effort as well as developer availability, with most of the techniques proposed tackling scheduling and staffing as an optimization problem. Many researchers have focused on using techniques found in the area of computational intelligence, as these have been proven to be extremely efficient for solving real-world problems that are large in size and high in complexity. The most common techniques include evolutionary algorithms (Alba and Chicano, 2007; Chang et al., 2008; Ren, Harman and Di Penta, 2011), fuzzy logic (Callegari and Bastos, 2009) and constraint satisfaction (Barreto, Barros and Werner, 2008). These have been adopted mainly due to their abilities to reduce problem search spaces and to model complex problems where there is a lack of mathematical analysis, as well as to effectively handle NP-hard problems (Chang et al., 2008).

The attempt presented here takes into account the non-interchangeable nature of human resources and aims to optimize assignments so that the level of developer experience is fully utilized, thus promoting quality software systems. Furthermore, swarm intelligence is investigated as a means to perform the optimization.

3 PROPOSED METHODOLOGY

3.1 Representation and Encoding

A software project comprises a number of tasks that must be performed in a predetermined sequence, with the dependencies between them satisfied at all times. Each task has a specified duration and requires developers to possess a set of skills in order to perform it. It is a project manager's responsibility to construct the project and form the development team and, in order to help project managers achieve this, a particle swarm optimization (PSO) algorithm was adopted. PSOs are a computational intelligence technique inspired by biological evolution occurring in nature (Eberhart and Kennedy, 1995). Swarm particles denote a candidate solution to the problem, and in this attempt the same representation defined in Stylianou and Andreou (2011) is used. Each particle's dimension uses mixed-type encoding to hold scheduling information and the assigned developers. Scheduling information is expressed by each task's starting day and the team staffing information is represented by a binary vector, where each bit shows whether or not a developer has been assigned to a task.

3.2 Particle Evaluation

The evaluation of each candidate solution p_i is assessed based on two factors, as shown in Eq. (1): (a) the computation of the degree of satisfaction of hard constraints and (b) the calculation of its fitness using objective functions.

$$eval(p_i) = constraint(p_i) + fitness(p_i) \tag{1}$$

where $constraint(p_i)$ and $fitness(p_i)$ denote the computed values regarding the hard constraints and objective functions, respectively. The first factor is used to assess the feasibility of a solution, whereas the second factor shows its quality.

A candidate solution is considered feasible if and only if it satisfies the imposed constraints, as shown in Eqs (2)-(4). Each constraint contains a penalty coefficient with a negative value in order to stress the existence of a violation.

$$c1(t) = penalty_{c1} \times \#\, violated\_days_t \times \#\, successors_t \tag{2}$$

$$c2(t) = penalty_{c2} \times \frac{\#\, unsatisfied\_\, skills_t}{\#\, required\_\, skills_t} \tag{3}$$

$$c3(e) = penalty_{c3} \times \frac{\#\, conflicting\_\, days_e}{\#\, working\_days_e} \tag{4}$$

Constraint $c1$ measures if there are violations of task dependencies, since it is required that each task's starting day must be set after all of its predecessors have completed. Constraint $c2$ measures if all skills required by a task are fulfilled by the developers assigned, since if the team does not possess one or more required skills then the task will not complete successfully and defects could occur. Constraint $c3$ measures if conflicts arise when developers are assigned to tasks, as they are not permitted to work on more than one task at any given time. The final constraint value of a particle is the summation of the individual constraint terms.

The fitness of a solution is evaluated using the two objective functions in Eqs (5) and (6). The former considers the duration of the project and the latter takes into account the experience of the assigned developers.

$$f_{duration} = \frac{\sum_{t=1}^{T} delay(t)}{T} \tag{5}$$

$$f_{skills} = \frac{\sum_{t=1}^{T} \left(\frac{\sum_{s=1}^{K} experience(s_t)}{K} \right)}{T} \tag{6}$$

The objective function $f_{duration}$ aims to schedule tasks so that there are no needless (idle) delays within the project, thus minimizing its overall duration. On the other hand, objective function f_{skills} aims to ensure that the teams will be the most suitable for the accomplishment of each task, and uses each assigned developer's degree of experience in the skills required. Since the two objectives are directly competing, it is often likely that the algorithm's attempt to increase one objective would cause the other to lower. Therefore, a trade-off mechanism using weights for each objective function, shown in Eq. (7), was implemented to allow software project managers to decide which of the two objectives is more significant for them.

$$fitness(p_i) = w_1 \times f_{duration}(p_i) + w_2 \times f_{skills}(p_i) \qquad (7)$$

where $0 < w_1, w_2 < 1$ and $w_1 + w_2 = 1$.

4 EXPERIMENTAL RESULTS

4.1 Design of Experiments

Initially, a survey was conducted with a number of software development SMEs in Cyprus in order to find out the driving factors influencing the size and complexity of a software project. With the information obtained, a total of 7 projects of varying size and complexity were used aiming to represent real-world software project case studies. The factors taken into account and their respective values in each project are provided in Table 1. Furthermore, three different sets of ratios for the weight values w_1 and w_2 (Eq. (7)) were used: equal importance (1:1), importance to project scheduling (9:1) and importance to developer experience (1:9).

4.2 Parameters and Execution

A combination of Constriction-PSO and Binary-PSO (Poli, Kennedy and Blackwell, 2007) variations were selected as the most suitable. Also, due to the multimodal nature of the problem having many global/local minimum, a low-connected ring topology was used so the swarm could adequately examine the search space and avoid premature convergence in local optimal solutions. The swarm size was kept constant at 60 particles and all 7 projects were executed 10 times for each weight ratio variation, with a maximum 10^6 number of iterations. In case that stagnation was observed, a partial re-initialization of positions and velocities took place. Finally, the penalty values for the constraints in Eqs. (2)-(4) were specified to -100.

4.3 Results and Discussion

As previously mentioned, the primary objective of this research attempt is to carry out an initial investigation as to whether the proposed approach produces acceptable solutions within the context of software project management. Therefore, each particle in the swarm was assessed, firstly, based on whether it represents a feasible software project schedule and developer assignments and, secondly, based on its ability to generate optimal solutions. The results of the executions are presented in Table 2. For the first project, all the final particles at the end of the algorithm's executions represent feasible solutions (since its feasibility rate equals 100%) and in addition all of them are optimal solutions (with a 100% hit rate). As the complexity and size of the software projects increase however, these percentages begin to decrease. Despite this, the algorithm always generates solutions that are feasible (but not necessarily optimal) even in the most complex and difficult project instances (i.e., 5 to 7). This indicates that the algorithm is highly capable of constructing adequate solutions with respect to the hard constraints imposed.

With respect to the quality of the produced solutions, the hit ratios in Table 2 show that the algorithm performs sufficiently well with the first four projects for all weight ratio variations. Here, the hit ratio percentages reach a maximum value of

Table 1: Software projects used to study the particle swarm optimization algorithm.

Project	Number of Tasks	Number of Dependencies (Rate)	Average Number of Skills per Task	Number of Available Developers	Average Number of Skills per Developer
1	10	13 (29%)	2	10	2
2	14	16 (18%)	2	10	1.5
3	18	24 (16%)	2	10	1.2
4	18	24 (16%)	2	5	0.7
5	25	15 (5%)	2.5	8	1
6	30	62 (14%)	3.3	18	2
7	30	62 (14%)	3.3	10	1

Table 2: Average feasibility and hit ratio percentages for each project for each weight ratio variation.

Weight Ratios	Average Feasibility Rate (%) | Hit Ratio (%)						
	1	2	3	4	5	6	7
1:1	100 | 100	99.1 | 30.0	97.9 | 50.0	96.6 | 30.0	89.0 | 0.0	88.2 | 0.0	85.0 | 0.0
9:1	100 | 100	99.6 | 50.0	97.8 | 40.0	95.6 | 30.0	88.8 | 0.0	87.3 | 0.0	83.8 | 0.0
1:9	100 | 100	98.0 | 90.0	96.0 | 80.0	95.0 | 70.0	90.0 | 0.0	89.0 | 0.0	87.0 | 0.0

100% in the first project but as the complexity increases, a progressive decrease is observed reaching as low as 30% in the fourth project. A possible explanation for the behaviour of the algorithm is that it encounters more difficulties when trying to satisfy the constraints since, intuitively, the fewer the number of available developers, the more likely that assignment conflicts will arise. With regards to projects 5 to 7, the algorithm experiences some difficulties in finding optimal solutions, despite being able to frequently generate feasible solutions (within 80%-90% of the time). This can suggest that the large increase in the complexity and size of software projects causes difficulties in the evolution of the algorithm and consequently to the generation of optimal solutions.

5 CONCLUDING REMARKS

The results obtained from various executions of the algorithm indicated that PSO is a promising approach for software project scheduling and team staffing, which performs sufficiently well in the majority of the projects examined in this paper. The average feasibility ratio of the solutions generated is more than 83% proving that most of the particles in a swarm reside in feasible search space area. However, some difficulties were encountered in the cases with larger-sized and more complex software projects, where the number of tasks, the type of dependencies and the number of available developers were shown to influence the ability of the algorithm to produce optimal solutions. Specifically in certain instances, the existence of “needless” gaps in project schedules was observed, despite satisfying all constraints. In order to increase the quality of solutions, an adjustment can be made to the objective functions so that they can more adequately handle gaps or by introducing new objective functions that could assist the swarm during its evolution. Furthermore, due to the obvious conflicting nature of the present objective functions, an implementation of a multi-objective version of the algorithm may perhaps be able to produce better results. These abovementioned adjustments are scheduled for future work along with experimentation with real software projects, which is currently in process with the collaboration of local software SMEs for the provision of data.

REFERENCES

Alba, E. and Chicano, J. F., 2007. Software project management with GAs. *Inform. Sciences*, 177(11), pp. 2380-2401.

Barreto, A., Barros, M. d. O. and Werner, C. M. L., 2008. Staffing a software project: A constraint satisfaction and optimization-based approach. *Comput. Oper. Res.*, 35(10), pp. 3073-3089.

Callegari, D. A. and Bastos, R. M., 2009. A multi-criteria resource selection method for software projects using fuzzy logic. In *11th International Conference on Enterprise Information Systems*. Milan, Italy, 6-10 May 2009. Berlin, Germany: Springer-Verlag.

Chang, C. K., et al., 2008. Time-line based model for software project scheduling with genetic algorithms. *Inform. Software Tech.*, 50(11), pp. 1142-1154.

Eberhart, R. and Kennedy, J., 1995. A new optimizer using particle swarm theory. In *6th International Symposium on Micro Machine and Human Science*. Nagoya, Japan, 4-6 October 1995. Piscataway, NJ, USA: IEEE Industry Applications Society.

Poli R., Kennedy J., Blackwell T., 2007. Particle swarm optimization: an overview. *Swarm Intelligence*, 1(1), pp. 33–57.

Ren, J., Harman, M. and Di Penta, M., 2011. Cooperative co-evolutionary optimization of software project staff assignments and job scheduling. In *International Symposium on Search Based Software Engineering*. Szeged, Hungary, 10-12 September 2011. Berlin, Germany: Springer-Verlag.

Standish Group, 2009. Standish Group CHAOS Report. Boston, MA, USA: Standish Group International, Inc.

Stylianou, C and Andreou, S. A. 2011. Intelligent Software Project Scheduling and Team Staffing with Genetic Algorithms. In *7th IFIP Conference on Artificial Intelligence Applications and Innovations*, Corfu, Greece, 15-18 September 2011, Berlin, Germany: Springer-Verlag.