A PSO Approach for Software Project Planning

Ya-Hui Jia[a,c], Wei-Neng Chen[b,c] (Corresponding Author) and Xiao-Min Hu[d]
[a]Department of Computer Science, Sun Yat-sen University
[b]School of Advanced Computing, Sun Yat-sen University
[c]Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education
[c]Engineering Research Center of Supercomputing Engineering Software, Ministry of Education
[d]School of Public Health, Sun Yat-sen University, P.R. China
chenwn3@mail.sysu.edu.cn

ABSTRACT
Search-based software project management is a hot research point in software engineering. Based on the event-based scheduler (EBS) we have proposed in previous work [1], this paper intends to further propose a two-phase particle swarm optimization approach which uses a set-based representation for task scheduling and an integer representation for workload assignment scheduling to improve planning performance. Experimental results on 83 instances demonstrate the effectiveness of the proposed approach.

Categories and Subject Descriptors
D.2.2 [**Software Engineering**]: Design Tools and Techniques – *Computer-aided software engineering.*

General Terms
Algorithms, Management, Performance, Design, Experimentation.

Keywords
Two-Phase, PSO, EBS, Scheduling, Staffing, Software Project, Planning, SCLPSO

1. INTRODUCTION
In software engineering, making an efficient project planning is very important [2]. Software project is people-intensive and consisted by various techniques. Also proficiency of workers matters a lot. Hence software project planning is quite difficult. For estimating workload and cost, some outstanding models like COCOMO [3] have proved themselves in real-world usage. And there are also many models and methods proposed for project scheduling and human resource allocation, e.g. critical path method (CPM) [4], project management net (PM-Net) model [5]. But these methods and models are rigid in somewhere. Chang et al. proposed a time-line based model in 2008 [6]. In spite of the flexibility it brings, the search complexity is also much higher than before. Considering the defects of these methods and models, previously we have developed a new scheduling strategy named even-based scheduler (EBS) with an ant colony optimization (ACO) algorithm [1].

In this paper, we intend to further propose a two-phase PSO algorithm to solve the software project scheduling and staffing problem based on EBS. The reason of using the PSO approach is as follow. First, set-based PSO was proposed to solve discrete problems and experimental results shows that set-based comprehensive learning PSO (S-CLPSO) is better than ant colony system (ACS) and some other ACO algorithms in some problems [7]. As the search space of task scheduling problem was originally represented by set, using S-PSO is promising. Second, the project planning problem involves two parts with different types of search space. As ACO is originally designed for discrete optimization problem, it is difficult to deal with the workload assignment. Two-phase PSO approach uses two different PSO algorithms aiming to these two different parts, S-PSO for task scheduling and traditional PSO for workload assignment so that both problems can be addressed effectively.

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.
ACM 978-1-4503-2881-4/14/07.
http://dx.doi.org/10.1145/2598394.2598422

2. MODEL DESCRIPTION
There are four main models used in this research, employee, task, objective and event-based scheduler.

1) Employee is the paramount resource in software project. Part of the goal in project planning is staffing workers properly. We describe the employee in three aspects, including wage, working hours and skills' proficiency. Different kinds of workers' payments are calculated by different formulas based on their working time and wage.
2) Scheduling tasks efficiently is the other part of goal in project planning. In this paper, the task schedule is described as a task precedence graph (TPG) [6]. Each task has five attributes, workload in person-months, required skills, number of people, deadline and penalty (if delayed). Using COCOMO model [3], we can calculate the task achievement monthly.
3) The total payment and penalties are considered as the expenditure. As for time consumption, we also evaluates it in money cost because employees must get payment every month. So the objective function is defined as the total expenditure of money.
4) Correspondent to the two problems in software project planning, a task list for task scheduling and an employee allocation matrix for human resource allocation are combined as whole solution in EBS. Schedule and staffing adjustment just occur when events happen such as project beginning, task finishing, and employee joining or leaving. The EBS was designed as two main parts. One is to assign workloads in chronological order. The other is local refinement which is used to choose suitable workers for each task.

3. THE TWO-PHASE PSO APPROACH
According to the representation scheme in EBS, a feasible plan includes two components: the task list and the planned employee workload assignment matrix.

1) The task list is a sequence of tasks. The search space of this sequence can be described by an N-tuple (ET^1, ET^2, …, ET^N), where ET^i is the set of tasks which can be placed at the i-th position of the sequence. A task list task list Y is feasible if and only if the sequence (Y^1, Y^2, …, Y^N) satisfies the precedence constrains.

2) The planned employee workload assignment matrix is sparse. Hence to improve the efficiency of construction for the matrix, we assign exact numbers of employees each task needs. Then for each assigned employee, we add the planned workload of this employee to the task. The possible workload of each assigned employee is an integer within [1, *mhpm*] (*mhpm* means max working hour per month) and the search space of this matrix can be described by an N-dimension matrix [EW^1, EW^2, …, EW^N], where EW^i is a vector consisted by all related employees' workloads. A solution Z is feasible if and only if the matrix [Z^1, Z^2, …, Z^N] satisfies some constrains such as the number of people in a task cannot exceed the max number of people needed in this task.

The whole solution consists of Y and Z denoted as $X=(Y, Z)$. We use S-CLPSO to get Y and CLPSO to get Z. Specific steps are as follow:

1) Initialization: For position initializing, the task list component Y can be obtained by randomly building a sequence of tasks which satisfies the precedence constraints. The planned workload assignment matrix Z can be built by randomly choosing numbers of employees and assigning random workload within [1, *mhpm*] for each selected employee. For velocity initializing, VY is an N-tuple with random possibility. Also VZ is same size matrix as Z. Each element in VZ is initialized as an integer within [1, $0.2\times mhpm$].

2) Velocity updating: For k-th particle, the velocity updating rules of VY and VZ are as follow:

$$VY_k^j \leftarrow \omega VY_k^j + cr^j(PBestY_{f_k(j)}^j - Y_k^j), \quad j = 1,2,...,n \qquad (1)$$

$$VZ_k^j \leftarrow \omega VZ_k^j + cr^j(PBestZ_{f_k(j)}^j - Z_k^j), \quad j = 1,2,...,n \qquad (2)$$

All operations in the formula (1) are same as defined in [7], and operations in formula (2) corresponds to the operations defined in [8].

3) Position updating: When update the task list, each particle converts its velocity VY_i^j into a crisp set $cut_\alpha(VY_i^j)$ firstly. Then the particle learns from the elements in $cut_\alpha(VY_i^j)$ to build the task list. In the S-CLPSO for software project planning, the selection methods are heuristic based on MINSK (minimal total slack time). Some other heuristic informations are available in [9]. We keep the CLSPO algorithm unchanged in updating the planned workload assignment matrix.

4) Local refinement: To further improve the performance, we use a local refinement strategy proposed in [1] called local mutation. In this strategy, system generates *mutate_num* neighbors by mutating the best-so-far solution in each end of iteration. If there is a better one in *mutate_num* mutated neighbors than the solution PSO found, then the mutated neighbor will become the new best-so-far solution.

4. EXPERIMENTAL ANALYSIS

Three real instances and eighty randomly generated instances are tested to prove the efficiency of our approach. All the random instances' TPG are derived from PSPLIB so that random instances have practical significance. And we compares the two-phase PSO approach with ACO proposed in [1]. For each instance, 30 attempts are made using both two approaches. Then we get the best results and the mean results for each instance. The proposed PSO approach yields the best results on 75 out of the 83 instances, also gets better mean results in 75 of the 83 instances. This comparison on expenditure indicates that not only the PSO approach does a better job than ACO in most instances but also it brings more stability. The results demonstrate that the proposed two-phase PSO is promising.

5. CONCLUSION

In this paper, a new approach, two-phase PSO, is proposed to solve the software project scheduling and staffing problem with the event-based scheduler. This work explores a new application domain for the S-PSO method and experimental results indicate that it is feasible to use PSO in this domain.

Though this research gains satisfying results now, there is still room to be better. To get further improvement, details in algorithm like parameters, selecting method, can be adjusted to meet the actual situation. In future research, it will be promising and significant to apply this approach to more human-centric project planning problems and people-intensive activities.

6. ACKNOWLEDGMENTS

This work was supported in part by the NSFC No.61379061, No.61309003, No.61202130, and No.61379060, in part by NSFC Joint Fund with Guangdong under Key Projects U1201258 and U1135005.

7. REFERENCES

[1] W.-N. Chen and J. Zhang. Ant Colony Optimization for Software Project and Staffing with an Event-Based Scheduler. *IEEE Trans. Software Eng.*, 39(1):1-17, March 2012

[2] L. L. Minku, D. Sudholt, and X. Yao. Improved Evolutionary Algorithm Design for the Project Scheduling Problem Based on Runtime Analysis. *IEEE Trans. Software Eng.*, to be published.

[3] B. Boehm et al. *Software Cost Estimation with COCOMO II*. Prentice-Hall. 2000.

[4] A. Shtub, J.F. Bard, and S. Globerson. *Project Management: Processes, Methodologies, and Economics*, second ed. Prentice Hall.

[5] C. K. Chang and M. J. Christensen. A Net Practice for Software Project Management. *IEEE Software*, 16(6):80-88, November 1999.

[6] C. K. Chang, H. Jiang, Y. Di, D. Zhu, and Y. Ge. Time-Line Based Model for Software Project Scheduling with Genetic Algorithms. *Information and Software Technology*, 50:1142-1154, 2008.

[7] W.-N. Chen, J. Zhang, H. Chung, W.-L. Zhong, W.-G. Wu and Y.-H. Shi. A Novel Set-Based Particle Swarm Optimization Method for Discrete Optimization Problems. *IEEE Trans. Evolutionary Computation*, 14(2):278-300, April 2010.

[8] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar. Comprehensive Learning Particle Swarm Optimizer for Global Optimization of Multimodal Functions. *IEEE Trans. Evolutionary Computation*, 10(3):281-295, June 2006.

[9] W.-N. Chen, J. Zhang, H. Chung, R.-Z. Huang, and Ou Liu. Optimizing Discounted Cash Flows in Project Scheduling—An Ant Colony Optimization Approach. *IEEE Trans. Systems, Man, and Cybernetics-Part C: Application and Reviews*, 40(1), January 2010.