A STABILITY AND EFFICIENCY ORIENTED RESCHEDULING APPROACH FOR SOFTWARE PROJECT MANAGEMENT

Yujia Ge
Department of Computer Science and Technology, Zhejiang Gongshang University, Hangzhou, China

Lijun Bai
Department of Logistics and Management, Zhejiang Gongshang University, Hangzhou, China

Keywords: Rescheduling, Genetic Algorithm, Software Project Management.

Abstract: Rescheduling gains more attention in recent years by researchers who focus their study on scheduling problem under uncertain situations. But in software engineering circumstances, it has not been widely explored. In this paper we propose a GA-based approach for rescheduling by applying a multi-objective objective function considering both efficiency and stability to produce new schedules after managers take control options to catch up their initial schedules. We also conducted case studies by simulation data. The results show the effectiveness of the rescheduling method in supporting decision making in a dynamic environment.

1 INTRODUCTION

During software project executions, changes seem to be inevitable which cause schedule slip from the initial plan. Ineffectual project control is the main cause of project over budget and behind schedule (Lederer, 1995). Software project managers feel the difficulties intrinsic to taking decisions especially when a large team of engineers work together. So an effective monitoring and rescheduling system is needed to support control decision making.

Recently rescheduling techniques are proposed in the areas, such as job shop problems (Pfeiffer, 2006; Cowling and Johansson, 2002). Nevertheless there is still very limited support under software projects circumstances. Sukhodolsky uses discrete optimization techniques for finding optimal control actions the manager should take to meet project's deadline (Sukhodolsky, 2001). But it is not practical as the situations in real projects may not be as simple as it is stated in the paper. Padberg applies simulated-based techniques for schedule optimization (Padberg, 2005). But the computational effort for the exact optimization of a Markov decision process in general grows exponentially with the size of the problem and the model currently can not fit in realistic problems.

An ideal situation for software managers is: when any factors change or status becomes bad to projects, the task assignment can be automatically updated by software management tools. So first, a model is constructed which is suitable to software project management and realistic problems. Then computing with reasonable complexity should be made to generate optimal control strategies.

Based on a relative simple model of software project management, a rescheduling approach is proposed based on Genetic Algorithms, In general, these scheduling or rescheduling problems are NP complete. GAs provides better solution to the complex project management problems than some other methods (Chang, 2001). Our previous models include task-based model (Chang, 2001) and capability-based model (Ge, 2006). In this paper, we further extend our work to facilitate control decision making.

In summary, this paper reports the following work.

(1) Modeling software project scheduling;

(2)Applying Genetic Algorithms on rescheduling with considering both efficiency and stability;

Ge Y. and Bai L. (2007).
A STABILITY AND EFFICIENCY ORIENTED RESCHEDULING APPROACH FOR SOFTWARE PROJECT MANAGEMENT.
In *Proceedings of the Second International Conference on Software and Data Technologies - SE*, pages 162-167
DOI: 10.5220/0001337301620167

(3) Conducting case studies.

2 UNCERTAINTY OF SOFTWARE PROCESS AND RESCHEDULING STRATEGIES

Two common rescheduling strategies are known: dynamics scheduling strategies and predictive-reactive strategy (Pfeiffer, 2006). In this paper, we focus on predictive-reactive strategies. Predictive-reactive strategy includes periodic, event-driven and hybrid. When a specified event occurs during schedule execution which has significant affect on the further execution, the schedule must be revised.

Such an event in software project can be turnover of essential personnel, technical approaches that may not work, unavailability of development or testing equipment and facilities (Kitchenham, 1997). Successful project control is based on the tracking of actual execution versus the project plan. Traditional EVA (Earned Value Analysis) techniques can be adopted for comparing a project plan and its execution. When a threshold reaches, rescheduling happens.

3 OUR APPROACH

3.1 The Model

In our software project scheduling model, a project is represented as a Task Precedence Graph (TPG).

- A TPG is an acyclic directed graph consisting of a set of tasks V={T_1, T_2, …, T_n} and a set of task precedence relations P={ (P_{ij}); i<>j, 1<=i<=n, 1<=j<=n}, where P_{ij} = 1 if T_i must be first completed before T_j can start, and zero if not.
- Associated with each task T_k is the estimated effort which can be obtained from any known estimation methods (such as COCOMO and COCOMO II) and required skills.

Resource modeling focuses on team members for a software project since they are major resources for any software project. Employees have a list of skills they possess, their salary rates, and their maximum workloads (a limit to the amount of work load they can be assigned). Proficiency levels of skills are not described since our previous research experiences suggest that too detailed modeling techniques lack of practical usage in dynamic situations. The combined skill set of the employees assigned to T_i (SK_EMP(i)) should include all the skills required by the task (SK_TASK(i)), i.e., $SK\_EMP(i) \supseteq SK\_TASK(i)$.

Efficiency

There are many performance objectives for project execution. The most common ones are minimum project duration and minimum total cost. Our efficiency measure is shown in Equation 1.

$$Efficiency = W1/OverLoad_{norm} + W2/Cost_{norm} + W3/Time_{norm} \quad (1)$$

OverLoad (minimum level of overtime) is the amount of time worked beyond the individual over time limits which is summed over all employees, and it is treated as a global objective for a project. $OverLoad_{norm}$ is computed by normalizing *Overload*. When applying GA in Section 3.2, $OverLoad_{norm}$ is achieved by dividing *overload* by the maximum overload in the population. *Similarly*, *cost* and *time* are also normalized as $Cost_{norm}$ and $Time_{norm}$. *Cost* (minimum cost) is the total labor cost of performing the project, computed using the labor rates of each resource and the hours applied to the tasks. *Time* (minimum of time span) is the total time span required to finish the project, from the start of the first task until the end of the last. *W1*, *W2* and *W3* are the weights chosen by managers for these three sub-objectives respectively.

Stability

When only considering efficiency measure of a schedule, a newly generated schedule will be often radically different from the initial one. Then the high cost of the changing staffing profile can not be neglected under this case. Practically speaking, managers are in favor of rescheduling strategies addressing continuity.

To minimize the impact of disruptions induced by new schedule, the performance measurement, stability, is applied. Two kinds of stability are recognized, i.e. ex post stability, ex ante stability (Herroelen, 2005). Ex post stability is considered here. There are several possible solutions for stability. Here Equation 2 (Pfeiffer, 2006) is adapted in our model to calculate stability in our objective function. The value is achieved by adding *starting time deviation* and *actuality penalty*.

$$\text{Penalty for Stability} = \left[\sum_{j \in B} \left(| t'_j - t_j | + k / \sqrt{t_j - T} \right) \right] / n \quad (2)$$

where B is the set of tasks that need to be rescheduled. They are the tasks that remained unprocessed in the initial schedule and still need to be processed under new circumstances. n is the number of tasks in B. t_j is the predicted start time of task j in the new schedule. t'_j is the predicted start of job j in the initial schedule . T is the current time.

The related objective function is shown in Equation 3.

$$Stability = W4/penalty\ for\ stability_{norm} \quad (3)$$

Penalty for stability$_{norm}$ is achieved by dividing *penalty for stability* by the maximum penalty for stability in the population

The composite objective function for evaluating a schedule considering efficiency and stability is shown in Equation 4.

$$The\ objective\ function = Efficiency * W_e + Stability * W_s \quad (4)$$

3.2 Stability and Efficiency Oriented Re-scheduling by Applying GA

Genetic algorithms which belong to stochastic search method have been widely used in many optimization fields and also provide good solutions in our previous research (Chang, 2001; Ge 2006). It remains our optimization method of choice.

The principle that using whatever encoding is the most natural to your problem and then devising a GA with that encoding has been widely accepted unless there is more theoretical progress on GAs. The genome here is an orthogonal 2D array with one dimension for task, the other for employee. The percent of an employee's labor that can be committed to any give task was constrained to a discrete set of values that are 0, 0.25, 0.5, 0.75 and 1.

Table 1 shows an example genome in which the number of row i and column j represents the commitment of Employee i to Task j. 0.25 means Employee 1 is assigned 25% of his working hours on Task 1.

Table 1: An example genome.

	T1	T2	T3	T4
Employee1	0.25	0	1	0.5
Employee2	0.5	0.5	0	0

When rescheduling is necessary, tasks and employees with changed profiles need to be updated first. Then the GA scheduler sets the initial schedule produced by preceding GA as the initial population of the new GA which takes advantage of previous result for the stability purpose. With the rows and columns of new employees or new tasks, genetic operator assigns each individual a specific value, for example, 0.25. The initializer results in a population that initially occupies only a small region of the total solution space but, if the GA propagation parameters are selected at all well, it will rapidly produce populations containing individuals with high figures of merit. Crossover operator is one-point crossover on 2D array and element flip is used as mutation operator.

Since genetic algorithms are non-deterministic, factors as population size, generation number, mutation probability and crossover probability not only influence the time required to perform the GA algorithm but also affect the quality of the result. The parameters for the proposed GA scheduler are determined after a set of test runs. The performance is better when the population size is bigger. But population size between 50 and 100 can still produce good results. Crossover probability is chosen between 0.4-0.8. Mutation probability is set between 0.01 and 0.1.

4 CASE STUDIES

Several simulation based tests were conducted. The settings for the GA are as follows: population size is 100, crossover probability is 0.4, and mutation probability rate is set to 0.01.

One case consists of 18 tasks for which 10 employees were available. Table 1 shows task properties. Employee properties are listed in Table 2.

Table 2: Task information table.

Task	PersonWeek	Skill	Ancestor
T0	15	0, 2	1, 2
T1	10	0, 2	7
T2	10	3, 4	3, 4
T3	10	1, 3	5
T4	15	1, 2	5, 8, 9
T5	10	2, 3, 4	6, 8
T6	20	0, 4	8
T7	10	1, 4	11, 14
T8	25	0, 2	14
T9	15	0, 3	10, 12
T10	20	2, 4	12
T11	10	2, 3	13
T12	10	3, 4	14
T13	10	0, 4	14
T14	20	1, 2	15
T15	20	1, 2	16
T16	20	1, 2	17
T17	20	1, 2	/

Table 3: Employee information table

Emp Id	Skill list	Salary/week
P1	0,1,3	1750.00
P2	0, 2	1000.00
P3	1	1000.00
P4	0, 3	1250.00
P5	1, 4	1250.00
P6	1, 3, 4	1500.00
P7	2, 4	1250.00
P8	1, 2, 3	1500.00
P9	0, 2, 4	1750.00
P10	2, 3	1250.00

The initial task assignment plan is shown in Table 4 and its corresponding Gantt graph is presented in Figure 1.

Table 4: Initial task assignment.

	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T16	T17
E1	1	0.5	0.8	0.5	1	0.5	0.8	0.5	0.3	0.8	0	0.3	0	0.3	1	0.5	1	1
E2	1	0.8	1	1	0	0.8	0.5	0.8	0.8	0.5	0.5	0	0.3	0.8	1	0.8	1	1
E3	0.8	0.3	0.5	0.5	1	0	0.5	0.3	1	0.8	0.8	0.8	0.8	0	1	1	0.3	0.5
E4	1	0.3	0.8	0.8	0.75	0.8	0.8	0.3	0.8	0	0.5	0.5	1	0.5	1	0.5	0.5	1
E5	1	0.8	0.3	0.8	0.5	0.8	0.8	0.5	1	0	0.8	1	0.5	0.5	0.5	0.8	1	1
E6	1	1	1	0	0.75	0.5	0.8	0.5	0.8	0.3	0.5	0.5	1	0.8	0.8	1	1	1
E7	1	0.8	1	0.3	1	0	0.8	0.8	1	0.8	0.8	0.5	0.8	0	0.8	1	1	0.8
E8	0.8	0	0.8	0.8	1	0.3	0.5	0	0.8	0.5	0.8	1	0.8	0.8	0.8	0.5	0.3	0.8
E9	0.8	0.3	0.5	0.5	1	1	1	0.3	1	0.8	0.3	0.3	0	0	0.8	0.8	0.5	1
E10	0.5	0.3	0.8	0.3	0.75	0.3	0.3	1	0.3	0.5	1	0.5	0.5	0.5	1	1	1	1

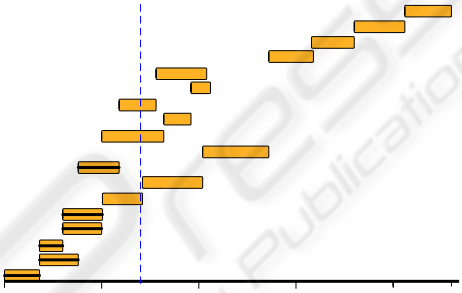

Figure 1: Initial schedule.

Suppose at the beginning of the 7th week T_0, T_1, T_2, T_3, T_4, T_7 have been completed and other tasks have not yet been initiated. The project is greatly behind the plan notified by tracking real execution data.

To bring any remaining project tasks into alignment with the planned schedule, managers have various project control actions to take, such as, adding people to the project, extending the time to completion, cutting out non-essential or less essential requirements. If option 1 is taken, genetic algorithm can easily generate a new schedule with the updated employees' information. If option 2 is used, it is just right-shift rescheduling and no genetic algorithm calculation needs to be done. If option 3 is taken, our model still easily fits by updating original task information tables.

The result going after option 2 is shown as follows. Suppose manager would like to add another engineer (as shown in Table 5) into this team to catch up the schedule.

Table 5: Newly added employee's information.

	Skill list	Salary/week
Employee 11	0, 2, 3	1250

Our rescheduling approach is applied for the remaining tasks that have not been started and Table 6 shows the comparison from the best newly generated schedule versus the initial schedule. t_{start} lists the start time of a specific task and *d* means the duration of this task. Rescheduling approach produced acceptable results both with stability (W_s=0, W_e =1) and without stability (W_s=1, W_e =1).

Table 6: Generated schedule versus initial schedule.

	Initial schedule		Results from Rescheduling (W_s=0, W_e=1)		Results from Rescheduling (W_s=1,W_e=1)	
	t_{start}	d	t_{start}	d	t_{start}	d
T5	5.03	2.11	7.00	1.38	7.00	1.54
T6	7.13	3.08	8.38	11.24	8.54	2.58
T8	10.21	3.33	11.24	2.63	11.12	2.5
T9	5.03	3.16	7.00	2.14	7.00	3.33
T10	8.19	3.48	9.14	3.33	10.33	2.16
T11	5.92	1.90	7.00	2.11	7.00	1.90
T12	11.67	1.82	12.48	1.25	12.50	1.29
T13	7.83	2.50	9.11	2.22	8.90	1.43
T14	13.54	2.35	13.87	2.00	13.79	2.11
T15	15.90	2.58	15.87	2.11	15.89	2.50
T16	18.48	2.67	17.95	2.00	18.39	2.76
T17	21.14	2.22	19.95	1.95	21.15	2.35
Time (week)	23.36		21.90		23.50	
Cost ($)	263664		264570		265234	

But the efficiency performance is affected by the stability measure. We can also see from the table that the schedule (W_s=0, W_e =1) has better efficiency performance over the schedule (W_s=1, W_e =1). Several other tests are conducted when control option 2 is taken. In all 4 cases, schedules without considering stability all have better performance as we expected. In case 1, 2 and 4, all of the generated new schedules without stability and with stability can be finished in or close to original deadline and budget. In case 3, generated schedule is far worse than it initial schedule which means managers may take other control options to reduce schedule slip for this situation.

Table 7: Comparison of schedules with stability versus schedules without stability

	No_{task}	No_{emp}	Intial schedule		Rescheduling (W_s=0, W_e=1)		Rescheduling (W_s=1,W_e=1)	
			Time (Week)	Cost ($)	Mean Time	Mean Cost	Mean Time	Mean Cost
1	5	5	7.52	98K	5.67	95K	7.23	99K
2	10	5	19.35	120 K	17.62	124K	18.12	131K
3	10	10	14.32	210 K	16.35	230K	16.42	232K
4	18	10	23.36	234 K	22.10	244K	23.67	266K

5 CONCLUSIONS AND FUTURE WORK

In this paper, we propose a software project rescheduling approach considering efficiency and stability. This approach is based on formulating software project scheduling and rescheduling situation as a multi-objective optimization problem via a genetic algorithm. The proposed method will help a manager to do scheduling with the option he made to put the project back on track. By case studies, the model can easily and efficiently do rescheduling under different kinds of control options.

Still, there are areas that can be improved. The parameters of stability and efficiency are determined in current research. Sensitivity study should be directed to balance the effect of stability and efficiency in different situations. In addition, evaluations of possible impact of all the available control options should be integrated in the model to support control decision making in software process.

REFERENCES

Cowling, P., & Johansson, M., 2002. Using Real Time Information for Effective Dynamics Scheduling, European J. of Operational Research, 139(2), 230-244.

Chang, C. K., Christensen, M., & Zhang, T., 2001. Genetic algorithms for project management, Annals of Software Engineering 11(1), 107-139.

Ge, Y., Chang, C. K. 2006. Capability-based Project Scheduling with Genetic Algorithms. CIMCA/IAWTIC 2006,161.

Herroelen, W., Leus, R., 2005. Project Scheduling Under Uncertainty: Survey and Research Potential, European J. of Operational Research, 165(2), 289-306.

Kitchenham, B., & linkman, S., 1997. Estimates, Uncertainty, and Risk, IEEE Software, 14(3),69-74.

Lederer, A. L., & Prasad, J., 1995. Causes Of Inaccurate Software Development Cost Estimates. Journal of Systems & Software, 31(2), 125-134.

Padberg, F., 2005. On the Potential of Process Simulation in Software Project Schedule Optimization, Proceeding of COMPSAC 2005, 127-130.

Pfeiffer, A. , Kadar, B., & Monostori, L., 2006. Stability-oriented Evaluation of Hybrid Rescheduling Methods in a Job-shop With Machine Breakdowns. In proceedings of MITIP2006 - 8th International Conference on The Modern Information Technology in the Innovation Processes of the Industrial Enterprises, 457- 462.

Sukhodolsky, J., 2001. Optimizing Software Process Control, Software Engineering Notes, 26(2), 59-63.