International Journal of Applied Science and Engineering
2007. 5, 2: 159-171

Robust Preemptive Resource Assignment for Multiple Software Projects Using Parameter Design

Lai-Hsi Lee*

Department of Information Management
National Ping Tung Institute of Commerce,
51 Min-Sheng E. Road, Pingtung, 900,Taiwan, R. O. C.

Abstract: Project preemption allows resources flow passively from one project or task to another. Due to high-cost and high-risk characteristics of software development, project preemption might have a significant impact on the ultimate completion time of a software project. This article examines the feasibility of the Taguchi's parameter design to achieve robust resource assignment for multiple software projects with different preemptive rules. Two examples of software projects is employed to demonstrate the design process of the parameter design applying in the project scheduling problems. The first project illustrates a preemptive single software project. Project preemption occurs when resource vacations are considered. The second example considers problems of preemptive multiple project scheduling. The parameter design with preemption rules and scheduling heuristics are employed to solve the problem. The results show that the parameter design can achieve robust resource assignment by minimizing the effect of task variation.

Keywords: Project Preemption; Robust Resource Assignment; Parameter Design.

1. Introduction

Solutions for solving a preemptive human resource assignment problem contain decisions for assigning the right task to the right person under the condition of allowing human resources flow passively from one project or task to another. Considering critical constrains of task complexity and resource scarcity, human resources for software projects are assigned to a certain scope of tasks by judging their technical skill suitable for the specialty and familiarity of the task, and therefore not easily substituted. When preemption occurs, a preemptive resource assignment problem for multiple software projects becomes more complex and needs to be solved robustly to reduce the impact of task complexity. Project resources in this paper are regard as human resources. Robust resource assignment aims to assign proper project resources under minimum turbulence of task variation, which means the selected resources can achieve project goals with most probability no mater how project tasks are over estimated or under estimated. Considering the difficulties for project managers to learn complex scheduling algorisms this paper aims to propose an easy and quick method for solving the preemptive resource assignment problem robustly.

Based on the definition by Liu and Cheng [1] preemptive scheduling problems are those in

* Corresponding author; e-mail: lhlee@npic.edu.tw *Accepted for Publication: December 05, 2007*

which the processing of a job can be temporarily interrupted and restarted at a later time. But what tasks should be re-scheduled among preemptive projects and non-preemptive projects? What impact there will be to the project objectives when project preemption occurs? And who is proper for the preemptive tasks? Considering those questions there are three issues that should be discussed further:

1. Project preemption: project preemption may occur in different ways. Project managers need to determine what type of preemption so as to identify which task should be preempted and which task should be interrupted.
2. Task priority: the priority of tasks should be pre-determined under the preemption rules. The determination of task priority requires easy to be operated and can be used to reduce the project duration.
3. Resource assignment methods: resource assignment methods focus on proper selection of resources. Considering the complex relationships among tasks, the method should be less sensitive to tasks variations.

Taguchi's parameter design is applied to solve the preemptive resource assignment problem since it is easy and famous for its robust designing results. This paper will propose the assignment process using Taguchi's parameter design for solving the preemptive problem. Two examples is employed to illustrate the use of the proposed method.

2. Literature review

2.1. Preemptive scheduling

Preemptive scheduling allows a task temporarily interrupted and restarted at a later time. Many of academy articles have discussed the scheduling problems, especial on the solutions of manufacturing processes. For example, Brasel and Hennes [2] consider an open-shop problem with *n* jobs being processed on *m* machines and preemption rule is each operation can be stop and continued later on; Averbakh and Xue [3] discuss the preemptive problem in a supply chain scenario with a customer and a manufacturer. Discussions of preemptive scheduling in project management are relatively few since complicate situations needed to be considered including preemption rules, resource assignment, project due date, respectively. Herroelen *et al*. [4] recommend that task preemption increases the computational complexity on the project resource scheduling problems. For software projects the problems have to consider further constrains non-renewable resources because programming skill are not easy to learn.

2.2. Project resource scheduling

Problems of project resource assignment had been wildly discussed for decades. Researchers propose various methods especially on resource constrained project scheduling problems (RCPSP) with different resource or task constrains. The use of linear algorithms is a main direction for solving the problems. For instances of recent development, Carlier and Neron [5] develop a linear method based on linear lower bounds to solve the traditional RCPSP. They [6] propose another algorithms with time-bound adjustment to compute redundant resources; Damay, Quilliot and Sanlaville [7] employ a linear programming model to solve preemptive and non-preemptive scheduling problems considering each task may or may not be preemptive. Other popular computational methods recently for solving the RCPSP are the application of genetic algorithms (GAs). For example, Alba and Chicano [8] use the GAs to calculate optimal resource scheduling for software project to reduce project budget; Ranjbar and Kianfar [9] employ the GAs incorporated with a local search method to achieve optimal resource utilization ratio.

Those algorithms might achieve optimal solutions under variety of constrains. However,

mass computation are required and not easy to understand become barriers for project managers to apply. Therefore, the advantage of easy use induces the heuristic methods play another important role for solving the scheduling problems. Scheduling heuristics are used to decide the priority of tasks among projects. Classical scheduling heuristics which are well known are First in First Serve (FIFS), Minimum Slack (MINSLK), and Shortest Activity From Shortest Project (SASP). FIFS is one of the most popular scheduling heuristic found in many multi-project environments, in which task priority is determined by its arrival time. Although MINSLK has many extension versions, the basic minimum slack time heuristic determines task's priority by its last starting time and earliest starting time. SASP was found to be effective in reducing mean completion time in a multi-project environment, in which task priority is based on its duration plus the project's remaining critical path time.

Many studies had been made to compare performances among these scheduling heuristics. Kurtulus and Davis [10] examined heuristic methods including MINSLK, SASP, and "maximum total work content." They concluded that SASP and "maximum total work content" are better than MINSLK. Dumond and Mabert [11] found that SASP is more effective in reducing mean completion time. Moreover, SASP and FIFS achieve better tardiness performance. Dumond [12, 13] conducted experiments to compare the performances of FIFS and SASP, specially to address the problem of projects with due date and limited resources. He concluded that FIFS performs well in reducing mean completion time, however, SASP completes small projects very quickly and delays large projects. Oguz and Bala [14] conducted computer experiments to test MINSLK and "shortest duration," and concluded that MINSLK achieves more number of best solutions while the "shortest duration" has a better solution quality on the average.

Scheduling heuristics had been proposed and modified for years to solve scheduling problems with different constrains. For recent development on issues of project scheduling, Yang, Geunes and O'Brien [15] develop a heuristic approach to balance of project costs of overtime and costs of tardiness. However, only one resource is considered in their model. Hartmann and Kolisch [16] and an update survey [17] summarize well-known heuristics for RCPSP and evaluate them with a standardized experimental test. Their collections of heuristics draw a clear picture of recent development of heuristics on RCPSP. However, the result is not able to point which is the best and new heuristics are proposed continuously. Such as Rabbani *et al.* [18] present a new heuristic using critical chain concept; Buddhakulsomsiri and Kim [19] propose a rule-based heuristic considering resource vacations and the split of project tasks. From paper review it is found that many heuristics are developed incorporated with other methods to achieve their research goals. In this paper, classical heuristics are employed incorporated with Taguchi's parameter design to present an easy operation process for solving the specific problem.

2.3. Taguchi's parameter design

Taguchi's parameter design is well known for its robust optimization [20]. The design is wildly applied in the field of engineering to select proper parameter levels on engineering design or manufacturing processes. For example, Liou, Lin, Lindeke and Chiang [21] employed the design to analysis the specification of robot kinematics parameters; Kim and Yum [22] try to find robust optimization of BPN parameter; Savas and Kayilci [23] use Taguchi's parameter design to select proper operation levels for a sand cast alloy.

Taguchi's parameter design [24] can be identified with two portions: an inner array containing the controllable factors and an outer array with noise factors. The controllable fac-

tors are the factors that can be monitored and are expected to select their proper levels so as to achieve higher performance. The noise factors are uncertainty factors that can not be well control. Those factors are regard as "noise" and are expected to minimum the effect affecting the performance. The objective of the outer array is to efficiently draw information of the joint effect of noise factors in order to achieve robust results. An inner array is designed to facilitate the optimization. These two special arrays allow Taguchi's parameter design to address the optimal conditions of controllable factors with minimum variation of noise factors. And the results of the experiment are regarded can achieve "robust" since the results is obtained by testing different conditions of noise factors.

The design using in issues of project management can be found in Santell *et al.*'s [25] report. They show the feasibility of parameter design apply in project management. Tsai, Moskowitz and Lee [26] make further applications on a resource selection problem for a single software project. And the parameter design will be employed in this paper to discuss multiple projects with preemptive tasks.

3. Proposed resource assignment process

Generally, Taguchi's parameter design has three main steps: (1) select factors and factor levels, (2) design experiment layout, and (3) conduct experiments and evaluate performance. Following those steps this paper presents the experimental process for resource assignment in software projects, including (1) measure factors of projects, (2) determine task priority, (3) design experiment layout, (4) evaluate project performance. The flow of the experimental process is illustrated as in Figure 1.

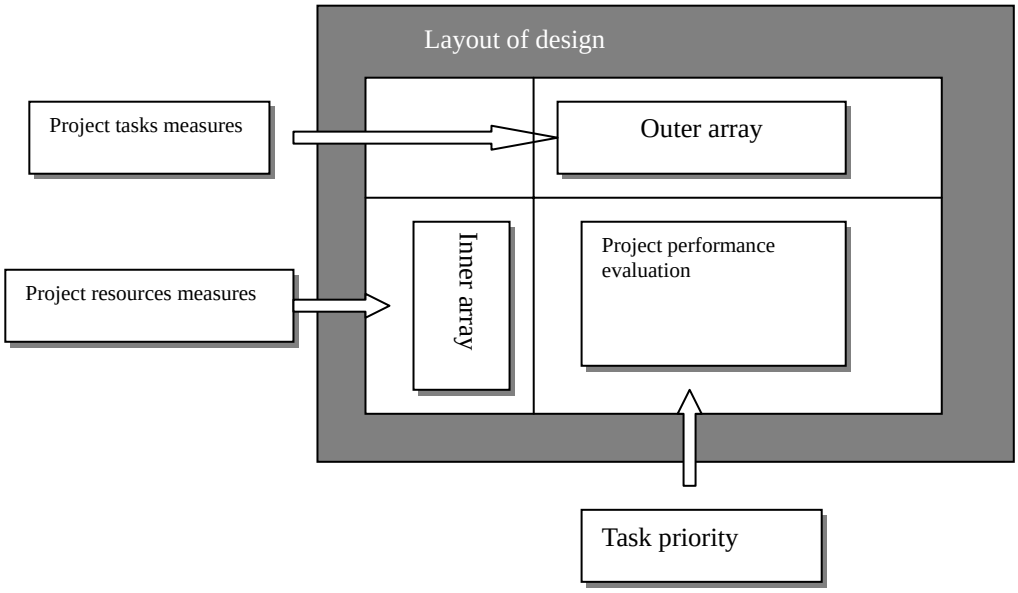


Figure 1. Experimental process of resource assignment.

3.1. Project task measures

Task complexity of software is not easily anticipated or measured and is thus treated stochastically. A software task often contains several subprograms (procedures, functions, or subroutines), thus its number of subprograms can measure the loading of a task by assuming the complexity of each subprogram being equal. Following three time estimates

on the program evaluation and review and review technique (PERT): optimistic (low complexity), normal (average complexity), and pessimistic (high complexity) estimates of the number of subprograms for each task are obtained to reflect three possible task complexity environments, denoted as levels 1, 2, and 3, respectively.

3.2. Project resource measures

Design capabilities and design costs are measured as two factors of resources. Resource capability is the average design speed, which is measured by the average design hours per subprogram. By experience, Resource's design hour is estimated as 5 hours per day. Resource cost is the average payoff per day.

3.3. Task priority

Task priority is pre-determined according to the design flow. Each software task has its precedence task until all tasks are completed. When multiple projects and preemption are considered the priority of tasks required re-scheduled by scheduling heuristics and preemption rules.

3.4. Project performance

In the experiment, total project costs with *n* projects and *m* resources are calculated by following formula:

$$PC = \sum_{i=1}^{n} (CT_i - DT_i) \times DC_i + \sum_{i=1}^{n} \sum_{j=1}^{m} (TL_{ij} \times RC_j), \quad (1)$$

where

PC = total project cost,
CTi = critical duration (days) of project i,
DCi = daily penalty for not complete project i on time,
DTi = deadline of the project i,
TLij = total duration (days) of resource j in project i,
RCj = daily payment of resource j.

4. Examples

4.1. Example 1: scheduling for a single project

A single software project [8] is employed to illustrate the process of the parameter design. There are five tasks (T1- T5) in the project with pre-determined priority as shown in Figure 2. Each task is estimated with task efforts. For example, Task T1 consists in creating the UML diagrams and is estimated 5 days to perform, denoted by $T_1^{effort} = 5$. Since the tasks T1 and T5 (testing) require collaborative work of all resources the two tasks are set to be one time estimates as the due date of the tasks. Task T2, T3 and T4 can be done by a single resource. Their task efforts are measured with three time estimates: optimistic, normal and pessimistic. For example, efforts of task T2 is $T_2^{effort} = \{17,20,23\}$, which is estimated 17 days in optimistic condition, 20 days in normal condition, and 23 days in pessimistic condition, respectively.

The work flow in Figure 2 shows two resources are required in this project, denoted as P1 and P2. Resource P1 and P2 will work together for T1 and T5, then resource P1 is arranged to design tasks of T2 and T4, resource P2 is sent to implement task T3.

There are two possible staffs for the project, denoted as R1 and R2. The work rate of Resource R1 is 1.2, which means R1 can work over time and the duration of a task is task effort divided by the correspondent resource's work rate. Resource R1's salary is $2000 per day, and will have vacation for two days every 10 days. Resource R2's work rate is 1.5, salary is $2500 per day, and will have vacation for two days every 7 days, respectively. When resources go for vacations the correspondent tasks will be temporally preempted until vacations are finish.

The scheduling problem of Example 1 is: which resource should be assigned as resource P1 who will be responsible for tasks T2 and T4, and which resource should be assigned as resource P2 who will implement task T3.

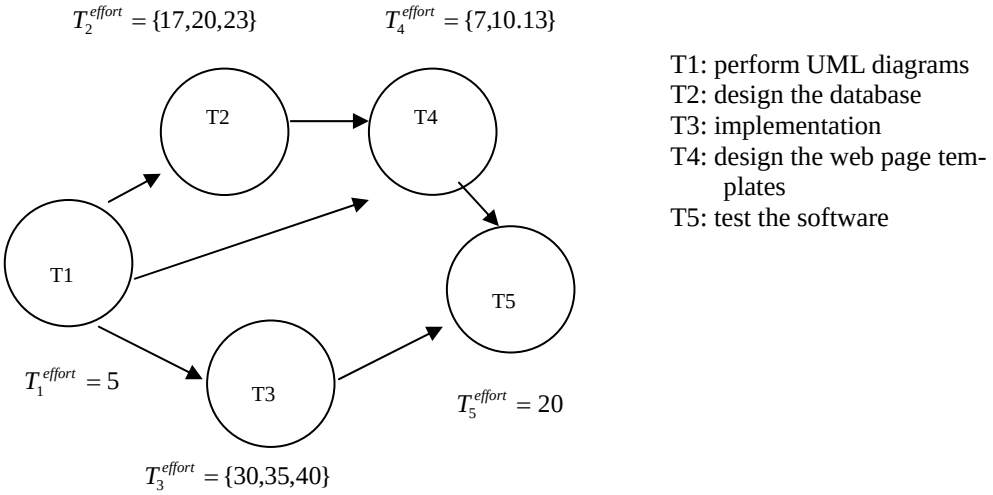


Figure 2. Work flow of Example 1

For the problem applying in the parameter design, the two candidates R1 and R2 are regarded as controllable resource levels. Two possible level combinations located in the inner array are {1, 2} and {2, 1}, presenting resource R1 as P1 and R2 as P2, or resource R2 as P1 and R2 as P1. Task complexities are regarded as levels of noise factors. The possible level combinations of the three tasks T2, T3 and T4 are 3^3=27 sets. To reduce the number of calculation this paper employs the L9 orthogonal array, which level combinations can be 9 sets. The related design layout is presented in Figure 3.

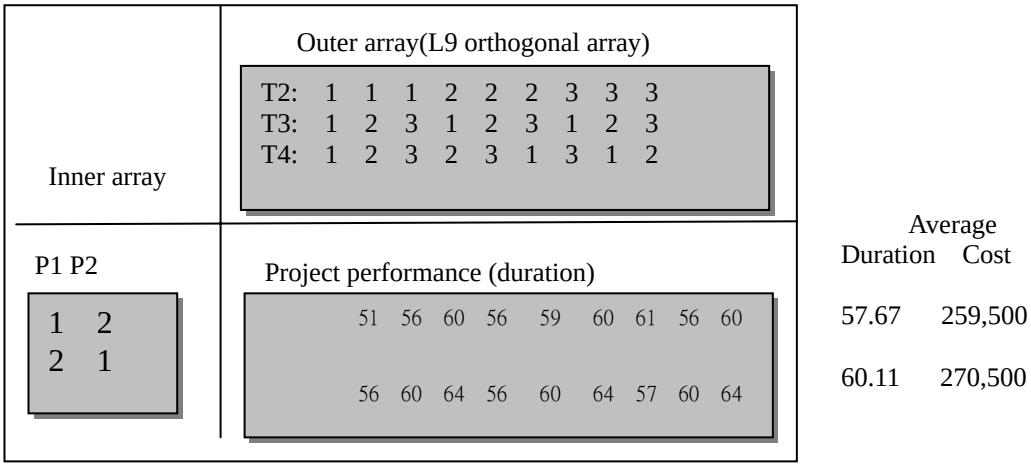


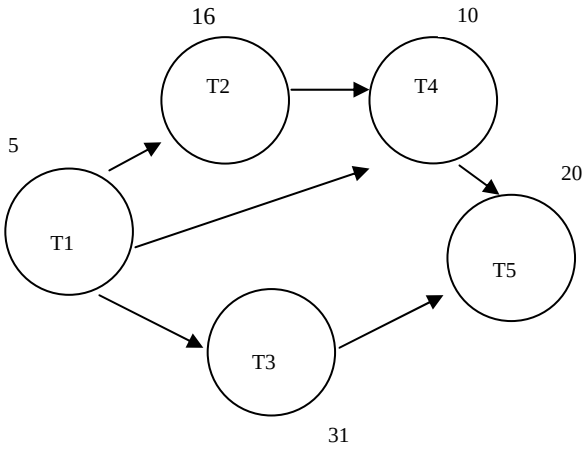
Figure 3. Results of Taguchi's parameter design of Example 1

Project performance is measured using project duration which is calculated from level

combinations of inner array and outer array. For example, the first level combinations of controllable factors and noise factors are {1, 2} for {P1, P2} and (1, 1, 1,) for {T2, T3, T4}, and task durations of T1 and T5 are set to be constant. The computation of project duration is listed in Table 1 and Figure 4. Table 1 shows the calculation process of task duration. The duration of T2, T3 and T4 are estimated by its work effort divided by the work rate of the correspondent resource, and plus the duration of resource vacation. For example, task duration of T2 is: 17 (work effort)/ 1.2 (work rate of correspondent resource R1) + 2 (days of vacation of resource R2= 16 days. After all task durations are estimated the project duration is then estimated by the rule of the critical path. For the example, the project duration is 51 days, and related project cost is (2000+2500)*51=229,500 (resource salary * project duration).

Table 1. An example of calculating task duration

	Controllable factors			Noise factors			
	P1	P2	T1	T2	T3	T4	T5
Level	1	2		1	1	1	
	Work rate 1.2	Work rate 1.5	Due date 5 days	Condition : Optimistic Work effort 17	Condition : Optimistic Work effort 35	Condition : Optimistic Work effort 7	Due date 20 days
Estimated duration				17/1.2=14 days	35/1.5=23 days	7/1.2=6	
Resource vacation				Int((5+14)/10)*2 =2 days	Int((5+23)/7)*2= 8 days	Int(5+14+7)/10-1)* 2=4	
Task duration				16	31	10	



Project duration: 51 days
Project cost: (2000+2500)*51=229,500

Figure 4. An example of calculating project duration and cost

The results of the parameter design in Figure 3 show the average project duration is 57.67 days and the average project cost is $259,500 when the level combination is {1, 2} of {P1, P2). Meanwhile, the duration is 60.11 days and the project is $270,500 when the level combination is {2, 1} of {P1, P2}. Therefore, it is suggested that resource R1 should be assigned to take tasks T2 and T4, and resource R2 should be assigned to take task T3.

4.2. Example 2: scheduling for multiple projects

4.2.1. Estimation of project parameters

In this example we consider a more complicate preemption problem. Two software projects from the Institute of Information Industry in Taiwan are examined. The first project, denoted as Project 1, was held earlier than the second project, denoted as Project 2. Project 1 had three human resources: R1, R2, and R3, each one being assigned with different tasks. Project 2 had shorter due date with higher penalty so that its tasks should be preempted to reduce project duration. Under the condition of no further resources being added, the project manager needed to decide who should be assigned to which tasks on Project 2.

According to the previous planning of system design, Project 1 is planned to have seven main tasks, denoted as A1 to A7, while Project 2 has 8 tasks, denoted as A8 to A13, respectively. The workflows of two projects are presented in Figure 5.

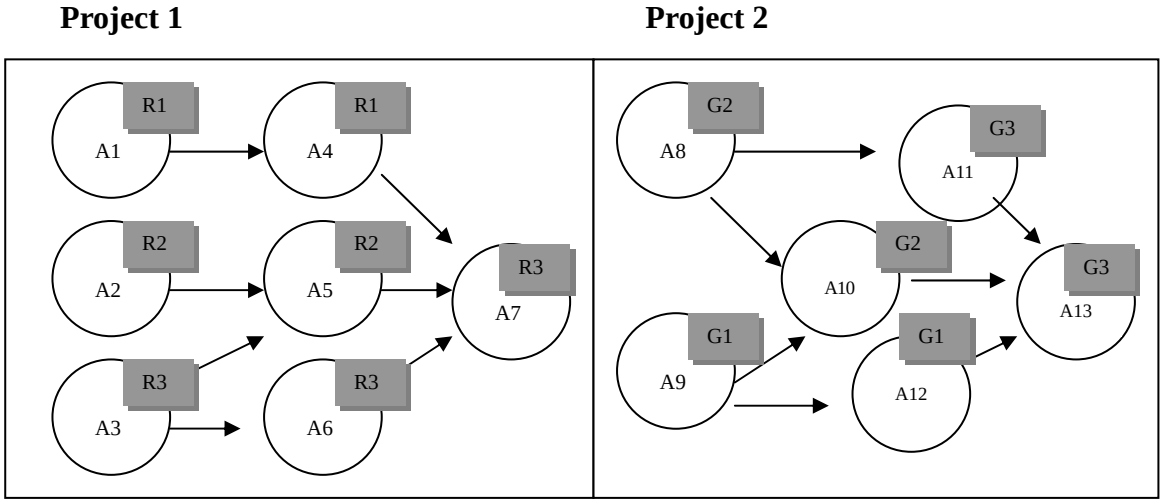


Figure 5. Project workflows of Example 2

Each node in Figure 5 refers to one software task and its corresponding human resource. Three human resources R1, R2 and R3 in Project 1 (the current project) were pre-determined and assigned with two or three tasks, for example, R1 worked on A1 and A4; R2 worked on A2 and A5; and R3 worked on A3, A6, A7, respectively. Since Project 2 would be preempted and no extra human resources are available, three current human resources have to be assigned to Project 2. Based on the specialty of design skill, all tasks in Project 2 can be categorized into three groups, denoted as G1, G2, and G3. Each group contains two tasks: G1 had A9 and A12; G2 had A8 and A10; G3 had A11 and A13. Due to work loading, each human resource is limited to one group only. Now, the problem is how to assign each human resource (R1, R2, R3) to which group (G1, G2, G3)?

Based on the contracts, the starting time of Project 2 is three days later than Project 1; the due date of Project 1 is 70 days, and the tardiness cost (penalty) is $10,000 per day after the due day, while the due date of Project 2 is 50 days and the penalty is $20,000 per day. There are 13 tasks Table 2 shows these estimates for 13 tasks in two projects.

The design speed and costs for three human resources are summarized in Table 3. The design speed of a resource is measured by the

average design hours per subprogram. On the average resource, the capability is 4, 6, and 5 hours per subprogram for R1, R2 and R3, respectively. By experience, Resource's design hour is estimated as 5 hours per day. Resource cost is the average payoff per day such as the daily payment of resource R1, R2, and R3 are $2000, $1500 and $1800, respectively.

Table 2. Optimistic, normal, and pessimistic estimates of the number of subprograms for each Task

Project 1					Project 2				
Task	Resource	Optimistic	Normal	Pessimistic	Task	Resource	Optimistic	Normal	Pessimistic
A1	R1	8	10	12	A8	G2	6	8	10
A2	R2	10	12	15	A9	G1	8	10	13
A3	R3	10	12	14	A10	G2	3	4	5
A4	R1	17	19	21	A11	G3	9	12	15
A5	R2	12	15	18	A12	G1	4	6	8
A6	R3	15	17	19	A13	G3	5	7	9
A7	R3	8	10	12					
Total		80	95	111	Total		35	47	60

Table 3. Design speed and costs of three human resources

Resource	Capability (Design hours/ subprogram)	Resource cost ($ per day)
R1	4	2000
R2	6	1500
R3	5	1800

4.2.2. Preemption scheduling

Bock and Patterson [27] summarized three project preemption rules: no preemption of resources (NPR), absolute priority for resources (APR), and limited priority for resources (LPR). NPR denotes resources are not preempted from any task, which provides a baseline for comparisons. APR has the absolute priority for assigning resources. When a preemptive project (or task) follows APR rule, resources are preempted from current projects (or tasks) without any constraint. A preemptive project (or task) under LPR rule receives higher but limited priority, in which resources are preempted only if the current task has positive slack time; otherwise the First In First Serve (FIFS) rule is applied to hold resources over current tasks.

The priority of tasks is determined by a preemption rule (NPR, APR, LPR) together with a heuristic (MINSLK, SASP, FIFS). Namely, if one project is preempted, the resource will be reallocated to the first task of the preempted project, and then the priorities of the rest tasks in two projects will follow one of three scheduling heuristics until all projects are completed. For example, if we use the APR rule together with FIFS and the resource R1 is assumed for group G2, then the priority of tasks will be A1-A8-A4-A10. This means that resource R1 should go for task A8 after A1 is completed since A8 has the absolute priority by the APR rule. After A8 is completed resource R1 should go back for A4 in Project 1 based on the FIFS heuristic.

4.2.3. Experimental design

The objective of the experiment is to assign

proper resources to the preempted project so as to optimize project performance. In the design, groups of tasks G1, G2 and G3 are regarded as controllable variables and resources R1, R2 and R3 are regarded as level 1, 2, and 3 for each variable. Since a human resource can take only one task group in Project 2, there are six possible combinations for reassigning three human resources to three task groups, say, {R1, R2, R3}, {R1, R3, R2}, {R2, R1, R3}, {R2, R3, R1}, {R3, R1, R2}, and {R3, R2, R1} to groups {G1, G2, G3}, respectively. These six possible combinations were arranged into the inner array in Figure 6.

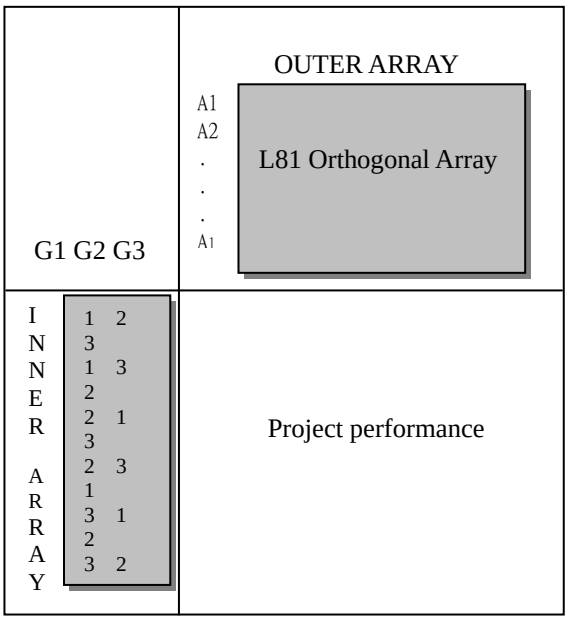


Figure 6. Design layout of Example 2

The outer array in Figure 2 contains 13 variables presenting the 13 tasks of the two projects. The levels of each noise factor are the three time estimates, where the optimistic, normal and pessimistic conditions are denoted as level 1, 2, and 3, respectively. The outer array is formed with 13 lines selected from lines 1, 2, 5, 10, 12, 13, 14, 27, 29, 31, 32, 33 and 34 of a L81 orthogonal array, where the selection rule is made by the first linear graph of the L81 array to avoid possible interactions (see e.g., Peace [24]). Considering all combinations of the inner and outer arrays, the design layout has a total of 6*81=486 different network trails, each with an easily computable project duration and cost.

4.2.4. Results

The proper resource assignment is determined by minimizing the project cost for each combination, as shown by the blocked entry in Table 4. For example, the minimum project cost is $534,944 for the assignment of {3, 2, 1} using NPR and FIFS. Namely, resource R3, R2, and R1 were reallocated to group G1, G2, and G3 in Project 2, respectively, and the resulting project cost is estimated as $534,944.

The conclusion findings in Table 4 are as follows:

1. The results showed that one project without a preemption rule might cause higher cost, for example, the average project costs for NPR are $710,170, $915,513 and $858,145, respectively, which are relatively much higher than others.
2. When FIFS is used, LPR is preferred since its project costs are relatively lower; for example, it's 539,794 of project cost. When MINSLK or SASP is used, APR is preferred.
3. Uniformly over all assignments, the combination of APR and SASP has the lowest project cost among all combinations. For example, its average project cost is $370,450 and the standard deviation is $1,833.
4. Under the combination of SASP and APR, the assignment of {1, 2, 3} achieves the overall minimum project cost of $367,298.

5. Conclusions

Project scheduling problems have been studied for decades; however, little attention has been paid on the methods for assigning proper

resources for multiple projects. This article employed the parameter design to achieve robust resource assignment quickly and less sensitive to task variations. Two examples of software projects are used to introduce the process of the parameter design applying in the preemptive multiple project scheduling problems. The first example illustrates scenarios of a single software project with two resources. Project preemption occurs when resources go for vacation and the project tasks will be interrupted until the vacation is finished. The problem of resource assignment is which resource should be assigned to take which tasks so as to obtain minimum project duration and project costs.

The second example employs the parameter design with different preemptive rules and scheduling heuristics to solve a scheduling problem of multiple projects. Although the scenario is complicated, the parameter design method shows its easiness of use and quick computation and it is feasible applying the method to the multiple project scheduling problems. Furthermore, the design can consider the complexities of project tasks and reduce the impact of high variations of project tasks so as to achieve a robust result. The scheduling problem can further consider the learning effect of human resources as the extent issue of the study. With the learning effect resources' design speed will be changed, and the result of the experiment might achieve different results.

Table 4. Comparison of total project costs for three preemptive rules and three heuristics

Assignment			FIFS			MINSLK			SASP		
G1	G2	G3	NPR	APR	LPR	NPR	APR	LPR	NPR	APR	LPR
1	2	3	827017	849337	723380	968419	371806	412668	1012107	**367298**	412668
1	3	2	634957	774244	567988	979636	377343	583712	905273	**371357**	582619
2	1	3	950820	955140	377383	923722	393535	424680	916168	**369080**	382349
2	3	1	574529	893136	416045	909604	375979	420927	749313	**371605**	419864
3	2	1	534944	820860	692788	871944	377645	419689	736361	**370420**	419689
3	1	2	738750	861300	461181	839752	384834	512658	829650	**372942**	482150
Average			710170	859003	539794	915513	380190	462389	858145	**370450**	449890
Std. Dev.			145569	56488	132776	49415	7098	64139	97279	**1833**	66356

References

[1] Liu, Z., and Cheng, T. C. E. 2002. Scheduling with job release dates, delivery times and preemption penalties. *Information Processing Letters*, 82: 107-111.

[2] Brasel, H., and Hennes, H. 2004. On the open-shop problem with preemption and minimizing the average completion time. *European Journal of Operational Research*, 157, (3): 607-619.

[3] Averbakh, I., and Xue, Z. 2007. On-line supply chain scheduling problems with preemption. *European Journal of Operational Research*, 181, (1): 500-504.

[4] Herroelen, W., De Reyck, B., and Demeulemeester, E. 1998. Resource-constrained project scheduling: a survey of recent development. *Computers and Operations*, 25, (4): 279-302.

[5] Carlier, J., and Neron, E. 2003. On linear lower bounds for the resource constrained project scheduling problem. *European Journal of Operational Research*, 149, (2): 314-324.

[6] Carlier, J., and Neron, E. 2007. Computing redundant resources for the resource constrained project scheduling problem. *European Journal of Operational Research*, 176, (3): 1452-1463.

[7] Damay, J., Quilliot, A., and Sanlaville, E. 2007. Linear programming based algorithms for preemptive and non-preemptive RCPSP. *European Journal of Operational Research*, 182,

(3): 1012-1022.
[8] Alba, E., and Chicano, J. F. 2007. Software project management with GAs. *Information Sciences*, 177: 2380-2401.
[9] Ranjbar, M. R., and Kianfar, F. 2007. Solving the discrete time/resource trade-off problem in project scheduling with genetic algorithms. *Applied Mathematics and Computation*, 191, (2): 451-456.
[10] Kurtulus, I., And Davis, E. W. 1982. Multi-project scheduling: categorization of heuristic rules performance. *Management Science*, 28: 161-172.
[11] Dumond, J., and Mabert, V. A. 1988. Evaluating project scheduling and due date assignment procedures: an experimental analysis. *Management Science*, 34, (1): 101-118.
[12] Dumond, J. 1992. In a multi-resource environment, how much is enough? *International Journal of Production Research*, 30, (2): 395-410.
[13] Dumond, E. J., and Dumond, J. 1993. An examination of resourcing policies for the multi-resource problem. *International Journal of Operations and Production Management*, 13, (5): 54-76.
[14] Oguz, O., and Bala, H. 1994. A comparative study of computational procedures for the resource constrained project scheduling problem. *European Journal of Operational Research*, 72: 406-416.
[15] Yang, B., Geunes, J., and O'Brien, W. J. 2004. A heuristic approach for minimizing weighted tardiness and overtime costs in single resource scheduling. *Computers and Operations Research*, 31, (8): 1273-1301.
[16] Hartmann, S., and Kolisch, R. 2000. Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 127, (2): 394-407.
[17] Kolisch, R., and Hartmann, S. 2006. Experimental investigation of heuristics for resource-constrained project scheduling: an update. *European Journal of Operational Research*, 174, (1): 23-37.
[18] Rabbani, M., Fatemi Ghomi, S. M. T., Jolai, F., and Lahiji, N. S. 2007. A new heuristic for resource-constrained project scheduling in stochastic networks using critical chain concept. *European Journal of Operational Research*, 176, (2): 794-808.
[19] Buddhakulsomsiri, J., and Kim, D. S. 2007. Priority rule-based heuristic for multi-mode resource-constrained project scheduling problems with resource vacations and activity splitting. *European Journal of Operational Research*, 178, (2): 374-390.
[20] Beyer, H. G., and Sendhoff, B. 2007. Robust optimization –a comprehensive survey. *Computer Methods in Applied Mechanics and Engineering*, 196, (33-34): 3190-3218.
[21] Liou, Y. H. A., Lin, P. P., Lindeke, R. R., and Chiang, H. D. 1993. Tolerance specification of robot kinematic parameters using an experimental design technique - the Taguchi method. *Robotics and Computer-Integrated Manufacturing*, 10, (3): 199-207.
[22] Kim, Y. S., and Yum, B. J. 2004. Robust design of multilayer feed forward neural networks: an experimental approach. *Engineering Applications of Artificial Intelligence*, 17, (3): 249-263.
[23] Savas, O., and Kayilci, R. 2007. Application of Taguchi's methods to investigate some factors affecting microporosity Formation in A360 aluminium alloy casting. *Materials and Design*, 28, (7): 2224-2228.
[24] Peace, G. S. 1993. *"Taguchi methods: a hands-on approach"*, Addison Wesley, Massachusetts, U.S.A.
[25] Santell, M. P., Jung, Jr., J. R., and Warner, J. C. 1992. Optimization in project coordination scheduling through application of Taguchi Methods. *Project Manage-*

ment Journal, 23, (3): 5-16.

[26] Tsai, H. T., Moskowitz, H., and Lee, L. L. 2003. Human resource selection for software development projects using Taguchi's parameter design. *European Journal of Operational Research*, 151: 167-180.

[27] Bock, D. B., and Patterson, J. H. 1990. A comparison of due date setting, resource assignment, and job preemption heuristics for the Multiproject Scheduling Problem. *Decision Science*, 21: 387-402.