J. Software Engineering & Applications, 2010, 3, 1125-1130
doi:10.4236/jsea.2010.312131 Published Online December 2010 (http://www.scirp.org/journal/jsea)

Weighted Multi-Skill Resources Project Scheduling

Fawaz S. Al-Anzi, Khaled Al-Zamel, Ali Allahverdi

College of Engineering and Petroleum, Kuwait University, Kuwait City, Kuwait
Email: Fawaz.Alanzi@ku.edu.kw, Alzamelk@eng.kuniv.edu.kw, allahverdi@kuc01.kuniv.edu.kw

Received October 13th, 2010; revised November 12th, 2010; accepted November 16th, 2010.

ABSTRACT

In this paper, we present an extension of the classical Resource Constrained Project Scheduling Problem (*RCPSP*). *We present a new type of resource constraints in which staff members are involved. We present a new model where staff members can have several skills with different proficiency, i.e., a staff member is able to perform more than one kind of activity as well as the time need is complete the task assign depends on the staff individual skill. We call this model the Weighted-Multi-Skill Project Scheduling Problem* (*WMSPSP*). *In our model, an activity has specific skill requirements that must be satisfied. To solve this problem, we propose a lower bound that uses a linear programming scheme for the RCPSP.*

***Keywords*:** *Weighted, Multi-Skill, Software, Project Scheduling, Lower Bound*

1. Introduction

The Resource Constrained Project Scheduling Problem (RCPSP) is a general scheduling problem [1]. It consists of a set of activities and a set of renewable resources. Each resource is available in a given constant amount. Each activity has duration and requires a constant amount of resource to be processed. Preemption is not allowed. Activities are related by two sets of constraints: temporal constraints modeled through precedence constraints and resource constraints that state that for each time period and for each resource, the total demand cannot exceed the resource capacity. The objective considered here is the minimization of the makespan (total duration) of the project. This problem is *NP-hard.*

Most work about RCPSP considers static problems in which activities are known in advance and constraints are fixed. However, every schedule is subject to unexpected events (consider for example a new activity to schedule, or a resource failure—e.g. machine breakdown).When such a situation arises, a new solution, taking these events into account, is needed in a preferably short time. Two classical methods used to solve such problems are: re-computing a new schedule from scratch each time an event occurs (a quite time consuming technique) and constructing a partial schedule and completing it progressively as time goes by (like in on-line scheduling problems—this is not compatible with planning purposes).Constraint Satisfaction Problems (CSP) are also increasingly used for solving scheduling problems.

The classical resource-constrained project scheduling problem (RCPSP) continues to be an active area of research attracting in recent years increasing interest from researchers and practitioners in the search for better solution procedures [2].

The RCPSP may be stated as follows. A project consists of a set of activities $V=\{1,\cdots,n\}$ where each activity has to be processed without preemption. The dummy activities 1 and n represent the beginning and the end of the project. The duration of an activity j is denoted by d_j where $d_1=d_n=0$. There are K renewable resource types. The availability of each resource type k in each time period is R_k units, $k=1,\cdots,K$. Each activity j requires r_{jk} units of resource k during each period of its duration where $r_{1k}=r_{nk}=0, k=1,\cdots,K$. All parameters are assumed to be non-negative integer valued. There are precedence relations of the finish-start type with a zero parameter value (*i.e.*, FS = 0) defined between the activities. $S_i\left(P_j\right)$ is the set of successors (predecessors) of activity j. It is assumed that $1\in P_i, j=2,\cdots,n,$ and $n\in S_j, j=1,\cdots,n-1$. The objective of the RCPSP is to find a schedule S of the activities, *i.e.*, a set of starting times $\left(S_1,S_2\cdots,S_n\right)$ where $S_1=0$ and the precedence

and resource-constraints are satisfied, in such a way that the schedule length $T(S) = s_n$ is minimized.

In this paper, we present an extension of the classical Resource Constrained Project Scheduling Problem (RCPSP). We present a new type of resource constraints in which staff members are involved. We present a new model where staff members can have several skills with different proficiency, *i.e.*, a staff member is able to perform more than one kind of activity as well as the time need is complete the task assign depends on the staff individual skill. We call this model the Weighted-Multi-Skill Project Scheduling Problem (WMSPSP). In our model, an activity has specific skill requirements that must be satisfied. To solve this problem we propose a lower bound that uses a linear programming scheme proposed for the RCPSP.

It is common in software engineering project to have several staff that can do several skills. The proficiency of every individual in each skill may vary. In this paper we present a Weighted-Multi-Skills Project Scheduling Problem (WMSPSP). This problem mixes both the Multi-Skill Project Scheduling Problem MSPSP [3], and the Multi-Purpose Machine model [4-6].MSPSP is a form of a Resource Constraint Project Scheduling (RCPSP) that uses the project description and adds new resource constraints inspired by the Multi-Purpose Machine model [3,7,8].For instance let us consider that resources are staff members having more than one skill, and that each activity needs a "given amount" of skill to be performed. Thus scheduling an activity at time *t*, requires matching its skill requirements with the skills of the staff members that are available at *t*. Our goal is to minimize the overall project, duration, *i.e.*, min (C_{max}).

The rest of the paper is divided into 5 sections. Section 2 is a formal presentation of the Weighted-Multi-Skill Project Scheduling Problem. Section 3 presents a case study of why the Weighted-Multi-Skill Project Scheduling Problem is different form classical Resource Constrained Project Scheduling Problem. Section 4 develops the lower bound of the problem. Finally, Section 5 presents the conclusions.

2. Weighted-Multi Skill Scheduling Problem

Figures 1((a),(b)) present a 4-activities and 4-members example with a feasible solution. **Table 1(a)** gives the processing times of activities along with their skill requirements. **Table 1(b)** describes staff members in terms of skills as it used to be given by MSPSP models. **Figure 1(a)** presents the precedence constraints between activities. **Figure 1(c)** shows a feasible solution.

The WMSPSP differs from MSPSP that **Table 2(b)** is modified to reflect that proficiency of a staff in certain skill in contrast with a plain binary value of Yes or No. This can be very true in real life situation especially with the escalating need of software engineering skills needs in the market place. Staff can have different experience and production rate using new emerging technologies and CASE tools. Notice that due to the different skills in the new model, a poor skilled person P_1 will need twice the time as normal skilled P_2 to finish a similar task of the type of Webmaster

In this example, one can see that activity A_3 cannot start at time 2, because it requires 2 programmers, while the available staff members at this time cannot meet this

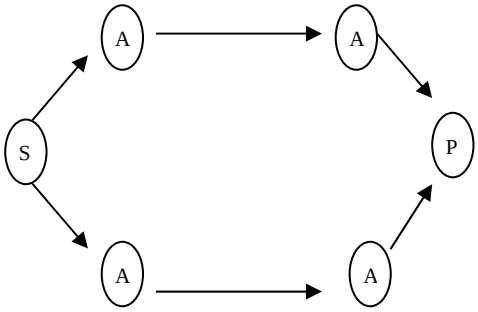

Figure 1 (a). Precedence constraints.

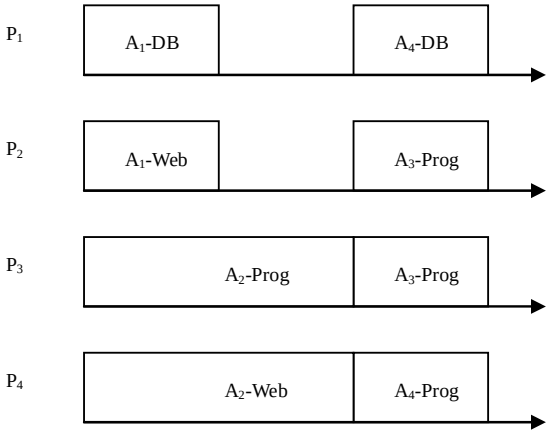

Figure 1 (b). Staff allocation time chart according to Weighted-Multi-Skill scheduling problem (WMSPSP).

Table 1 (a). Activity definition.

	A_1	A_2	A_3	A_4
Processing time	2	5	3	3
Programmer	-	1	2	1
DB Designer	1	-	-	1
Webmaster	1	1	-	-

Table 1 (b). Classical MSPSP person definition.

	P_1	P_2	P_3	P_4
Programmer	-	Yes	Yes	Yes
DB Designer	Yes	-	-	-
Webmaster	Yes	Yes	-	Yes

Table 1 (c). Proposed WMSPSP person definition.

	P_1	P_2	P_3	P_4
Programmer	-	1.0	1.0	0.7
DB Designer	1.0	-	-	-
Webmaster	0.5	1.0	-	0.7

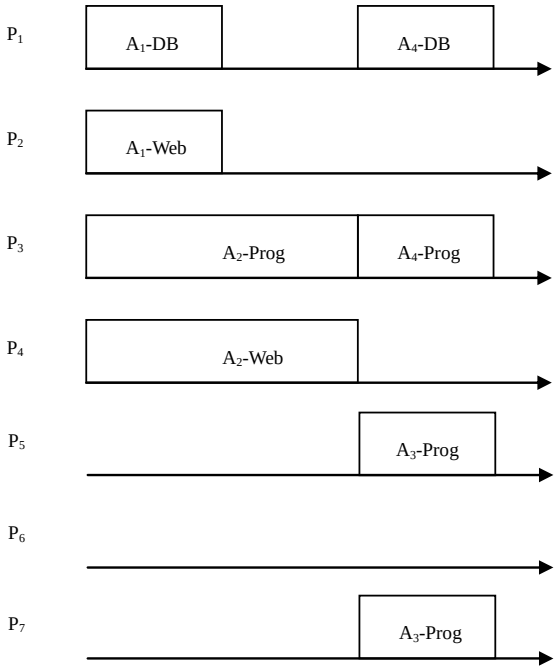

Figure 1 (c). Staff allocation time chart according to Weighted-Multi-Skill scheduling problem (WMSPSP) for Table 1(b).

skill requirement (P_1 is not a programmer).

This model is an extension of the classical RCPSP: if we assume that all members only have one skill with equal proficiency, we get classical resource constraints. This model can also be seen as a specific case of the Multi-Mode RCPSP [9]. The main reason to justify this new model is the huge number of modes (more than two hundred feasible modes for a medium size project) that would be necessary instance, consider activity A_2 of the example presented in **Tables 1(a), 1(b) and 1(c)**. This activity requires a Programmer (who can be P_2, P_3, P_4) and a Webmaster (who can be P_1, P_2, P_4). If we use a Multi Mode model in which persons are considered as resources, there are six valid modes for A_3: {(P_1, P_2), (P_2, P_4), (P_3, P_1), (P_3, P_2), (P_3, P_4)}. In the WMSPSP the processing time for the activity depends on the resource assignments change from a mode to another. Therefore, a decision maker that has to build a schedule should select the best modes for the project.

Let us present WMSPSP notations:

- $\{A_1, \cdots, A_n\}$: the set of activities to be processed without preemption.
- T_i: the processing time of A_i.
- $G = (V, E, d)$: the precedence graph in which there is a node (i) associated to each activity A_i. A starting dummy activity (s) and an ending dummy activity (p) are added. $(i, j) \in E$ if there is a precedence constraints between A_i and A_j, in that case d_{ij} which is the valuation of $(i, j) \in E$, is equal to P_i.
- $\{S_1, \cdots, S_K\}$: the set of skills.
- $\{P_1, \cdots, P_M\}$: set of staff members.
- $S_{m,k} > 0$ if person P_m has the skill S_k and 0, otherwise, $\forall m \in [1, \cdots, M]$ $\sum_k S_{mk} > 0$ indicates that a person has at least one skill.
- $b_{i,k}$: the number of normal skill persons with the skill S_k, needed to perform activity A_i,
- r_i: the release date of A_i is the longest path in G from the starting dummy activity (s) to the end of node (i).
- q_i: the tail of A_i is the longest path in G from the end of node (i) to the ending dummy activity (p), minus the processing time of A_i.

After this presentation of the Multi-Skill Project Scheduling Problem, we present a lower bound for this problem.

3. A Case Study

We will use the same example used in this paper but with different skill matrices for single and multi skill cases to show that solutions will yield different strategies or optimal schedule in each of the two cases.

Back to the activities definition table which shows constrains whether the required time or the skills needed to finish each activity.

The following table is the original table which has seven skills as total so **Table 2(b)** will be equivalent to **Table 1(b)** regarding the available resources.

The following table shows the weight of the skill for each person and applying the same concept to **Table 2(b)**, we will get **Table 1(c).**

Table 2 (a). Activity definition for MSPSP.

	A_1	A_2	A_3	A_4
Processing time	2	5	3	3
Programmer	-	1	2	1
DB Designer	1	-	-	1
Webmaster	1	1	-	-

Table 2 (b). Classical MSPSP person definition.

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
Programmer	-		Yes		Yes		Yes
DB Designer	Yes	-	-	-			
Webmaster		Yes	-	Yes		Yes	

Table 2 (c). Proposed WMSPSP Person Definition.

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
Programmer	-		1.0		1.0		0.5
DB Designer	1.0	-	-	-			
Webmaster		1.0	-	0.7		1.0	

Notice that one of the available resources is not being utilized at all and we face another if we put the total cost in our consideration.

4. A Lower Bound for WMSPSP

In the resource constrained project scheduling problem (RCPSP), non-preemptive activities requiring renewable resources, and subject to precedence constraints, have to be scheduled in order to minimize the makespan. RCPSP has been proved to be *NP-hard*. Thus a large amount of work has been devoted to the computation of lower bounds (LBs). Some of them are based on linear programming. Others relax the non-preemption constraint and associate a variable with each subset of activities that can be processed simultaneously without violating either resource or precedence constraints. So they take into account explicitly the resource requirements of activities. Consequently the linear program can be very large, but it can be tackled, either by column generation, or by solving heuristically a relaxation to a weighted node packing problem. Other bounds are based on the cumulative scheduling problem (CuSP), which is an extension of the *m*-machine problem where the resource requirements of activities can be larger than 1. A CuSP is obtained by choosing a resource and relaxing the constraints due to other resources.

The method used nowadays [10] to generate instances is a random procedure that guaranties that a specific characteristic has a specified value, so it is possible to generate several random instances diverse in the considered characteristics. This method does not give any clue about the optimal solution. Because the RCPSP is a NP-hard problem, there is no algorithm available today to calculate the optimal solution, unless the instance is small enough or is easy enough to solve. Having optimal solutions can be useful also for the study of the complexity of the instances. Without the optimal solution any indicator of the complexity of the instance will be inexact. There is no method for RCPSP that allows the generation of an instance with known optimal solution, for any instance size, and applicable in any network.

In [11], several approximation algorithms for the RCPSP are compared in the same machines, with equal time limits and an instance set including large instances. A priority rule is based on ranking the activities according to a criterion, and then building a schedule by giving priority to activities with a higher ranking over the activities with a lower ranking, but without violating precedence and resource restrictions. It is a simple technique for obtaining solutions, since the only thing required is to calculate a value for each activity.

The best result of the priority rules will be the first Upper Bound and the start point for the approximation algorithms, and only the improve made over this value will count for the performance of the algorithm in our new performance indicator. The priority rules used are: Latest Start Time (LST);

Latest Finish Time (LFT); Shortest Processing Time (SPT); Greatest Rank Positional Weight (GRPW); Most Total Successors (MTS); Most Total Successors Processing Time (MTSPT). The first two rules use the latest start and finish time calculated in the CPM, and the logic is the same, schedule first the activities that must start immediately or the project will be delayed.

The use of Lower Bound is also important, because an approximated algorithm only know that it has an optimal solution when it obtains a solution with equal value of the Lower Bound. Having a good Lower Bound avoids losing time searching for a better solution when the optimal solution was already found.

A simple instance is an instance that the optimal value can be obtained with only a simple priority rule. The implementation of the test is simple, calculates the Lower and Upper Bounds, and if they are equal, then the instance is simple. Next all the simple problems found in the instance set should be removed.

A good performance indicator is essential to interpret the results. Normally the mean percentage over the Upper or Lower Bound is used. If the optimal value is available, it is normally used, but for large instances normally there is no known optimal value. The problem with these indicators is that a meta-heuristic that does nothing, is classified, and there is no notion of the worst value that can be archived. Suppose that all meta-heuristics results are between 10 and 11 for an instance problem, and in other instance the meta-heuristics are all between 10 and 15. The result will penalise for the worst meta-heuristic in the first instance 10% ((11-10)/10), and in the second instance 50% ((15-10)/10). We think that all instances must value the same. The proposed performance indicator assigns the same weight to each problem, and does not consider problems where all algorithms archive the same result. The indicator formula is the following:

$$\text{Performance} = (1/N)\sum_{i=1}^{N}(R_i - V_i)/(R_i - UB_i)$$

In this formula R_i is the value of the starting solution (initial Upper Bound), V_i is the value of the solution ob-

tained for the meta-heuristic, and UB_i is the best value obtained for this problem with all meta-heuristics.

In [12], the authors have classified the multitude of heuristic procedures for the RCPSP with respect to their building blocks such as e.g. schedule generation scheme (SGS), meta-heuristic strategy, and solution representation. They have also tested the heuristics on three sets of benchmark instances from the PSPLIB library.

They also summarize recent heuristics from the literature. The approaches are grouped into priority rule-based X-pass methods; classical meta-heuristics namely genetic algorithms, tabu search, simulated annealing, and ant systems; non-standard meta-heuristics such as local search and population-based methods; and other heuristics like Forward–backward improvement (FBI).

They employ the three test sets which consist of 30, 60, and 120 activities, respectively. Each set has been generated using a full factorial design of parameters which determines the characteristics of the resource and precedence constraints. In total, they have 480 instances with 30 activities, 480 instances with 60 activities, and 600 instances with 120 activities. The instances have been used by many researchers, and they are available from the project scheduling library PSPLIB in the internet.

For the above reasons, it is important to develop acceptable lower bounds for NP-hard problem. In this section we present a lower bound for WMSPSP. Let's not forget that computing lower bounds for the WMSPSP is a challenging problem. Lower bounds are useful, first to prove the efficiency of heuristics, and eventually to be used in branch-and-bound methods. The lower bound that we propose is destructive [13] in the sense that it is used to determine if a given number LB is a valid lower bound for the project duration. Once LB is fixed, a deadline d_i, is associated to each activity ($d_i = LB - q_i$).

The linear bound that we present is an adaptation of a linear programming scheme proposed by Carlier and Neron [14] for the RCPSP and MSPSP proposed by Neron [3], which is based on time-horizon decomposition into successive intervals. The first step is the computation of time-intervals. We assume that release dates (r_i) and deadlines (d_i) have been computed according to the precedence constraints and a given integer *LB*. Let $R = U_{i\in 1}\{r_i, d_i\} = \{t_1, t_2, \cdots, t_{L+I}\}$. We assume that R is sorted in a non-decreasing order and that all time points are different. Let:

- $\forall 1 \in [1 \cdots L], \quad e_1 = [t_1, t_{1+1}]$. L denotes the number of consecutive time intervals that must be taken into account. e_1 is the 1-th interval and t_1 is the starting point of time-interval e_1.
- $\forall 1 \in [1 \cdots L], \forall i \in [1 \cdots I]$, $x_{i,l}$ is the absolute part of A_i performed during e_1. $x_{i,l}$ are variables for our linear program,
- $\forall 1 \in [1 \cdots L], \forall m \in [1 \cdots M], \forall k \in [1 \cdots K]$. $\delta^1_{m,k}$ the time a person P_m spent during e_1 performing skill S_k. $\delta^1_{m,k}$ are variables for our linear program.

The first constraint (1) implies that the parts of activities are positive:

$$\forall i \in [1 \cdots I], \forall 1 \in [1 \cdots L], X_{i,1} \geq 0 \tag{1}$$

The second constraint (2) ensures that the activities are completely performed:

$$\forall i \in I, \sum_{I=1}^{L} x_{i,1} = T_i \tag{2}$$

Constraints (3)-(5) are used to model that an activity must be performed within its time-window. Moreover, for each interval, the part of the activity performed during this interval must not be larger than the size of the interval itself. Notice that (4) and (5) are linear constraints since r_i, d_i and t_1 are known beforehand (they are data in our linear programming formulation)

$$\forall 1 \in [1 \cdots L], \forall i \in I, \quad x_{i,1} \leq t_{1+1} - t_1 \tag{3}$$

$$\forall i \in I, \forall 1 \in [1 \cdots L], \quad \text{if } d_i \leq t_1 \text{ then } x_{i,1} = 0 \tag{4}$$

$$\forall i \in I, \forall 1 \in [1 \cdots L], \quad \text{if } r_i \geq t_{1+1} \text{ then } x_{i,1} = 0 \tag{5}$$

The following four Equations (6-9) are used to model the resource constraints. Skill requirements of activities must be met for each interval (6). A time interval being given, a staff member cannot work longer than the size of this time-interval (7). A staff member can perform a given skill only if he has it (8). Total time spent on tasks must meet the task requirements (9).

$$\forall k \in [1 \cdots K], \forall l \in [1 \cdots L] \quad \sum_{i \in I} x_{i,1}.b_{i,k} \leq \sum_{m=1}^{M} \delta^1_{m,k} \tag{6}$$

$$\forall l \in [1 \cdots L], \forall m \in [1 \cdots M] \quad \sum_{k=1}^{K} \delta^1_{m,k} \leq t_{1+1} - t_1 \tag{7}$$

$$\forall l \in [1 \cdots L], \forall m \in [1 \cdots M], \forall k \in [1 \cdots K], \quad \delta^1_{m,k} \leq S_{m,k}.(t_{1+1} - t_1) \tag{8}$$

$$\forall k \in [1 \cdots K], \quad \sum_{l=1}^{L} \sum_{m=1}^{M} \delta^1_{m,k} = \sum_{i=1}^{n} b_{i,k} \tag{9}$$

Remember that deadlines are computed according to *LB* and the precedence graph. Then if there is no solution, there is at least one non-valid deadline, thus there is no solution with a makespan equal to *LB*.

5. Conclusions

In this paper, an extension of the classical Resource Constrained Project Scheduling Problem (RCPSP) was pre-

sented. It is a new type of resource constraints in which staff members are involved where staff members can have several skills with different proficiency, i.e., a staff member is able to perform more than one kind of activity as well as the time need is complete the task assign depends on the staff individual skill. We call this model the Weighted-Multi-Skill Project Scheduling Problem (WMSPSP). In this model, an activity has specific skill requirements that must be satisfied. We proposed a lower bound that uses a linear programming scheme for the classical RCPSP.

6. Acknowledgements

This Research is sponsored by Kuwait university Research Administration project number EO 01/07.

REFERENCES

[1] A. Elkhyari, C. Gueret and N. Jussien, "Solving Dynamic RCPSP Using Explanation-Based Constraint Programming," http://www.emn.fr/jussien/publications/elkhyari-MAPSP03.pdf

[2] V. Valls, F. Ballestín and S. Quintanilla, "Justification and RCPSP: A Technique That Pays," *European Journal of Operational Research*, Vol. 165, No. 2, September 2005, pp. 375-386. doi:10.1016/j.ejor.2004.04.008

[3] E. Neron, "Lower Bounds for the Multi-Skill Project Scheduling Problem," *Proceeding of the Eighth International Workshop on Project Management and Scheduling*, Spain, 2002, pp. 274-277.

[4] P. Baptiste, C. Le Pape and W. Nuijten, "Satisfiability Tests and Time Bound Adjustments for Cumulative Scheduling Problems," *Annals of Operation Research*, Vol. 92, No. 0, 1999, pp. 305-333 doi:10.1023/A:1018995000688

[5] S. Dauzere-Peres, W. Roux and J. B. Lassere, "Multi-Resource Shop Scheduling with Resource Flexibility," *European Journal of Operational Research*, Vol. 107, No. 2, June 1998, pp. 289-305. doi:10.1016/S0377-2217(97)00341-X

[6] B. Jurish, "Scheduling Jobs in Shops with Multi-Purpose Machines," Ph.D. Dissertation, University of Osnabrück, Osnabrück, 1992.

[7] R. Kolisch, "Serial and Parallel Resource-Constrained Project Scheduling Methods Revisited: Theory and Computation," *European Journal of Operational Research*, Vol. 90, No. 2, April 1996, pp. 320-322. doi: 10.1016/0377-2217(95)00357-6

[8] E. Neron, P. Baptiste and J. N. D. Gupta, "Solving Hybrid Flow Shop Problem Using Energetic Reasoning and Global Operations," *Omega*, Vol. 29, No. 6, December 2001, pp. 501-511. doi:10.1016/S0305-0483(01)00040-8

[9] A. Sprecher, "Resource-Constrained Project Scheduling, Exacts Methods for the MultiMode Case," Lectures Notes in Economics and Mathematical Systems, Springer Verlag, Berlin, 1994.

[10] J. S. Coelho, "Generating RCPSP Instances with Known Optimal Solutions," *Proceedings of PMS*, 2004

[11] J. S. Coelho and L. V. Tavares, "Comparative Analysis on Approximation Algorithms for the Resource Constrained Project Scheduling Problem," PMS2002, Eighth International Workshop on Project Management

[12] R. Kolisch and S. Hartmann, "Experimental Investigation of Heuristics for Resource-Constrained Project Scheduling: An Update," *European Journal of Operational Research*, Vol. 174, No. 1, October 2006, pp. 23-37. doi:10.1016/j.ejor.2005.01.065

[13] R. Klein, A. Scholl, "Computing Lower Bounds by Destructive Improvement: An Application to Resource-Constrained Project Scheduling," *European Journal of Operational Research*, Vol. 112, No. 2, January 1999, pp. 332-346. doi:10.1016/S0377-2217(97)00442-6

[14] J. Carlier and E. Neron, "On Linear Lower Bound for the Resource Constrained Project Scheduling Problem," *European Journal of Operational Research*, Vol. 149, No. 2, September 2003, pp. 314-324. doi:10.1016/S0377-2217(02)00763-4

