The Resource-Constrained Project Scheduling Problem: New Benchmark Results

Creemers S[1,2]

[1] IESEG School of Management, France
s.creemers@ieseg.fr
[2] KU Leuven, Belgium
stefan.Creemers@kuleuven.be

Keywords: project scheduling, RCPSP, lower bound.

1 Introduction

Over the past decades, the resource-constrained project scheduling problem (RCPSP) has become a standard problem in the operations research literature. The goal of the RCPSP is to schedule a set of project activities $V = \{0, 1, \ldots, n\}$ such that the makespan of the project is minimized, while satisfying precedence and resource constraints. Precedence constraints impose that an activity $i : i \in V$ can only start upon completion of all its predecessors $\mathbf{P}_i = \{j|(j,i) \in E\}$, where E is the set of precedence constraints. Resource constraints, on the other hand, impose that an activity i can only be scheduled if sufficient resources are available. There are K renewable resource types, and the availability of each resource type $k : k \in \mathcal{K} = \{1, 2, \ldots, K\}$ is denoted by R_k. Each activity i requires $r_{i,k}$ units of resource k. A solution to the RCPSP is a schedule $S = \{S_0, S_1, \ldots, S_n\}$, where S_i is the starting time of activity i. The project starts at time $S_0 = 0$, and completes at S_n, where activities 0 and n are dummy activities that represent the start and the completion of the project, respectively. In addition, define $\mathscr{A}(S,t) = \{i : S_i \leq t \wedge (S_i + p_i) \geq t\}$, the set of activities in schedule S that are active at time t, where p_i is the duration of activity i. Without loss of generality, we assume $p_i \in \mathbb{N}$ for all $i : i \in V$. The RCPSP can then be formulated as a combinatorial optimization problem:

$$
\begin{aligned}
\min \quad & S_n & & \\
\text{s.t.} \quad & S_i + p_i \leq S_j & & \forall (i,j) \in E \\
& \sum_{i \in \mathscr{A}(S,t)} r_{i,k} \leq R_k & & \forall t \geq 0, \forall k \in \mathcal{K} \\
& S_i \geq 0 & & \forall i \in V.
\end{aligned}
$$

Note that we assume that activities are executed without preemption, and that scheduling decisions are made at discrete points in time.

Blazewicz et al. (1983) have shown that the RCPSP is strongly $\mathscr{NP}$-hard, which explains the abundance of heuristic solution methods in the literature (refer to, e.g., Kolisch and Padman, 2001; Kolisch and Hartmann, 2006; Hartmann and Briskorn, 2010). In this abstract, however, we focus on exact methods. Among exact methods, the branch-and-bound (BB) procedures of Demeulemeester and Herroelen (1992, 1997), Mingozzi et al. (1998), and Sprecher (2000) are still the most successful approaches to solve the RCPSP. In what follows, we compare the performance of several lower bounds (LBs), and compare the performance of four BB procedures and the state-of-the-art procedure of Demeulemeester and Herroelen (1997).

2 Branch-and-bound procedures

A BB procedure is an iterative algorithm that implicitly enumerates the state space of a combinatorial optimization problem using a search tree. The root node of the tree corresponds to the original optimization problem, and its child nodes correspond to subproblems or feasible solutions (also called "leaf nodes"). Each iteration, a search strategy is used to select the active node from which new nodes (i.e., children) are generated using a branching scheme. If it can be shown (using dominance rules and/or bounds) that the optimal solution cannot be found by visiting the children of a node, that node is fathomed (i.e., its branch is pruned from the search tree). Eventually, if all nodes have been (implicitly) visited, the procedure stops, and the optimal solution is obtained as the best feasible solution found in any of the leaf nodes.

Several BB procedures for solving the RCPSP have been proposed in the literature. In most of these procedures, nodes correspond to partial schedules, and a depth-first search strategy is used to select the next active node. The procedures, however, use different branching schemes and pruning methods (i.e., dominance rules and LBs). In this abstract, we consider the following branching schemes:

- The precedence tree branching scheme proposed by Patterson et al. (1989), and later on adopted by Sprecher (1998).
- The delay alternatives branching scheme proposed by Christofides et al. (1987), and later on adopted by Demeulemeester and Herroelen (1992, 1997).

For other branching schemes, refer to Stinson et al. (1978), Mingozzi et al. (1998), and Igelmund and Radermacher (1983). Even though a depth-first search strategy is generally accepted as the best choice, in this abstract, we will also consider a breadth-first search strategy. This results in four BB procedures to be tested.

3 Dominance rules

Several dominance rules have been proposed in the literature. The state-of-the-art procedure of Demeulemeester and Herroelen (1992, 1997) uses the following dominance rules:

- DH1: if an activity i cannot be scheduled together with any other activity, it should be started at the first time possible.
- DH2: if a pair of activities i and j cannot be scheduled together with any other activity, they should be started at the first possible time possible.
- LLS: we can fathom a node that is associated with a (partial) schedule where an activity i can be scheduled 1 time unit earlier (i.e., where activity i can be left-shifted). A global variant of this local left shift rule can also be devised (see e.g., Schrage, 1970).
- CUT: we can fathom a node associated with (partial) schedule S' if we previously have saved another (partial) schedule S'' that dominates schedule S'. Note that Demeulemeester and Herroelen (1992, 1997) use a hash function to save (partial) schedules.

From these dominance rules, we will only consider CUT. In contrast to Demeulemeester and Herroelen (1992, 1997), however, we do not overwrite a schedule if a schedule is already stored at a given hash (i.e., we keep track of all schedules).

4 LBs

In the literature, we distinguish between two classes of LBs: (1) complex procedures that are used to obtain a very tight LB and (2) fast procedures that can be evaluated as

part of a subroutine in the nodes of a BB procedure. In this abstract, we focus on the latter class of LBs. Several of these LBs have been proposed in the literature (refer for instance to Brucker et al. (1999, 2003), Klein and Scholl (1999), Knust (2015), and Coelho and Vanhoucke (2018)). In this abstract, we consider the following LBs:

- CPL: the critical path LB.
- CS: the critical sequence LB that was introduced by Stinson et al. (1978).
- LB3: the node-packing LB as implemented by Demeulemeester and Herroelen (1997). This bound ranks activities in a list based on the number of "companions" each activity has (i.e., the number of activities with which it can be scheduled in parallel). To construct the LB, activities (and their companions) are removed from the list.

Next, we propose two new LBs:

- LB4: an extension of LB3 that recalculates the list of activities each time an activity (and its companions) is removed from the list.
- LBO: an overflow LB that determines the overflow (i.e., the remaining work contents) of activities that cannot be fully scheduled in a schedule that uses the critical path as a baseline; the remaining work contents is scheduled after the critical path.

The performance of the different LBs is evaluated for the instances of the well-known J30, J60, J90, and J120 PSPLIB data sets (Kolisch and Sprecher, 1996). For each data set, Figure 1 reports how often a LB is one of the dominant LBs. Conversely, Figure 2 reports how often a LB is uniquely the dominant LB. From the results it is clear that LBO performs very well, especially if larger projects are considered.

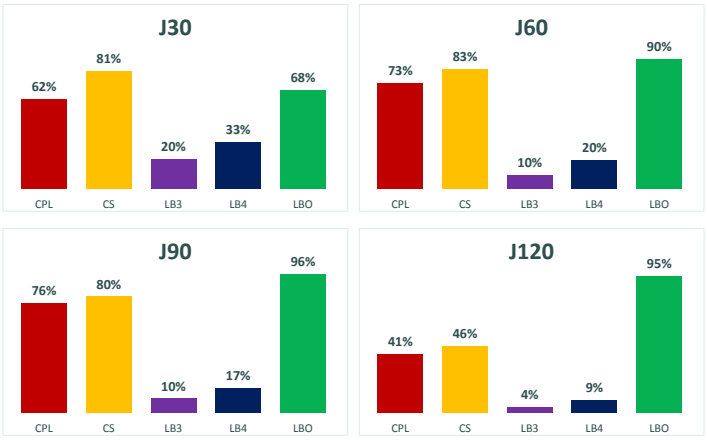

Fig. 1. How often is a LB the best LB

5 Results

In this section, we compare the performance of the procedure of Demeulemeester and Herroelen (1997) (as implemented in the Rescon software; see also Deblaere et al. (2009)) and four other BB procedures:

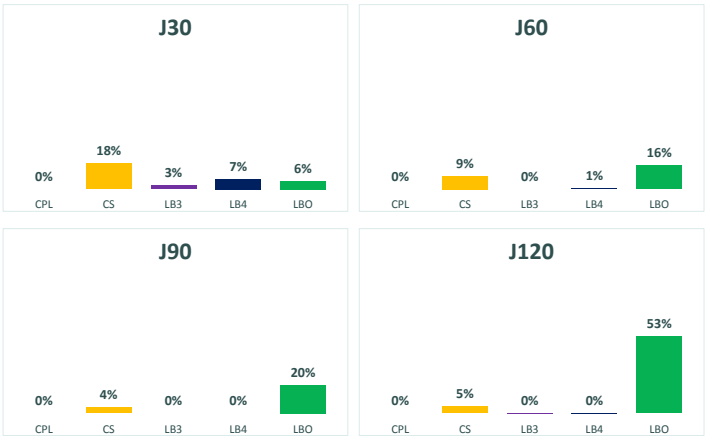

Fig. 2. How often is a LB better than any other LB

- Precedence tree branching scheme and depth-first search strategy.
- Precedence tree branching scheme and breadth-first search strategy.
- Delay alternatives branching scheme and depth-first search strategy.
- Delay alternatives branching scheme and breadth-first search strategy.

Each of these BB procedures is equipped with the CUT dominance rule, and with LBs CPL, CS, LB3, LB4, and LBO. In addition, the procedures that adopt a breadth-first search strategy evaluate a trivial upper bound in each node. We use these procedures to solve all instances of the J30 PSPLIB data set. In order to make a fair comparison, all tests are performed on the same computer: an Intel I5 2.6GhZ computer with 8GB of working memory.

The result of the preliminary test are reported in Table 1. From the table it is clear that we easily outperform the procedure of Demeulemeester and Herroelen (1997). In addition, the results also show that, in contrast to popular belief, a breadth-first search strategy almost performs as good as a depth-first search strategy.

Table 1. Benchmark results for procedure of Demeulemeester and Herroelen (1997) and various other procedures that are equipped with dominance rule CUT and LBs CPL, CS, LB3, LB4, and LBO

		Precedence Tree		Delay Alternatives	
	DH1997	Depth First	Breadth First	Depth First	Breadth First
# Nodes	19.30E6	167.14E6	14.48E6	11.66E6	6.36E6
CPU (sec)	374	1185	209	207	149

We have also performed a number of other preliminary tests that use new dominance rules and LBs that have not been discussed in this abstract. The results of these experiments have shown that we can solve all instances of the PSPLIB J30 data set in 27 seconds (while

visiting 6.15E6 nodes). If we minimize the number of visited nodes, we end up with 1.29E6 nodes over all instances of the J30 data set (with a CPU time of 183 seconds). Compared to the state-of-the-art procedure of Demeulemeester and Herroelen (1997), this results in a improvement of factor 13.8 for CPU times and of 14.8 for memory requirements.

References

Blazewicz J., Lenstra J.K. and Rinnooy Kan A.H.G., 1983, "Scheduling subject to resource constraints: Classification and complexity", *Discrete Applied Mathematics*, Vol. 5, pp. 11–22.

Brucker P., Drexl A., Möhring R., Neumann K. and Pesch E., 1999, "Resource-constrained project scheduling: Notation, classification, models, and methods", *European Journal of Operational Research*, Vol. 112, pp. 3–41.

Brucker P., Knust S., 2003, "lower bounds for resource-constrained project scheduling problems", *European Journal of Operational Research*, Vol. 149, pp. 302–313.

Christofides N., Alvarez-Valdes R. and Tamarit J.M., 1987, "Project scheduling with resource constraints: A branch and bound approach", *European Journal of Operational Research*, Vol. 29, pp. 262–273.

Coelho J., Vanhoucke M., 2018, "An exact composite lower bound strategy for the resource-constrained project scheduling problem", *Computers and Operations Research*, Vol. 93, pp. 135–150.

Deblaere F., Demeulemeester E. and Herroelen W., 2009, "RESCON: Educational project scheduling software", *Computer Applications in Engineering Education*, Vol. 19, pp. 327–336.

Demeulemeester E., Herroelen W., 1992, "A branch-and-bound procedure for the multiple resource-constrained project scheduling problem", *Management Science*, Vol. 38, pp. 1803–1818.

Demeulemeester E., Herroelen W., 1997, "New benchmark results for the resource-constrained project scheduling problem", *Management Science*, Vol. 43, pp. 1485–1492.

Hartmann S., Briskorn D., 2010, "A survey of variants and extensions of the resource-constrained project scheduling problem", *European Journal of Operational Research*, Vol. 207, pp. 1–14.

Igelmund G., Radermacher F.J., 1983, "Preselective strategies for the optimization of stochastic project networks under resource constraints", *Networks*, Vol. 13, pp. 1–28.

Klein R., Scholl A., 1999, "Computing lower bounds by destructive improvement: An application to resource-constrained project scheduling", *European Journal of Operational Research*, Vol. 112, pp. 322–346.

Kolisch R., Sprecher A., 1996, "PSPLIB: A project scheduling problem library", *European Journal of Operational Research*, Vol. 96, pp. 205–216.

Kolisch R., Padman R., 2001, "An integrated survey of deterministic project scheduling", *Omega*, Vol. 29, pp. 249–272.

Kolisch R., Hartmann S., 2006, "Experimental investigation of heuristics for resource-constrained project scheduling: An update", *European Journal of Operational Research*, Vol. 174, pp. 23–37.

Knust S., 2015, "Lower bounds on the minimum project duration". *in Schwindt and Zimmermann, Eds. Handbook on Project Management and Scheduling: Vol. 1*, Springer: Heidelberg, pp. 43–55.

Mingozzi A., Maniezzo V., Ricciardelli S. and Bianco L., 1998, "An Exact Algorithm for the Resource-Constrained Project Scheduling Problem Based on a New Mathematical Formulation", *Management Science*, Vol. 44, pp. 714–729.

Schrage L., 1970, "Solving resource-constrained network problems by implicit enumeration: Nonpreemptive case", *Operations Research*, Vol. 18, pp. 225–235.

Sprecher A., 2000, "Scheduling Resource-Constrained Projects Competitively at Modest Memory Requirements", *Management Science*, Vol. 46, pp. 710–723.

Stinson J.P., Davis E.W. and Khumawala B., 1978, "Multiple resource-constrained scheduling using branch-and-bound", *AIIE Transactions*, Vol. 10, pp. 252–259.

Patterson J.H., Słowiński R., Talbot F.B. and Węglarz J., 1989, "An algorithm for a general class of precedence and resource constrained scheduling problems". *Advances in Project Scheduling*, Elsevier: Amsterdam, pp. 3–28.