Resource-constrained project scheduling: A survey of recent developments

Willy HERROELEN

Erik DEMEULEMEESTER

Bert DE REYCK
Singapore Management University, bdreyck@smu.edu.sg

Resource-constrained project scheduling A survey of recent developments

Willy Herroelen • Erik Demeulemeester • Bert De Reyck
Department of Applied Economics, Katholieke Universiteit Leuven (Belgium)

Abstract

Resource-constrained project scheduling involves the scheduling of project activities subject to precedence and resource constraints in order to meet the objective(s) in the best possible way. The area covers a wide variety of problem types. The objective of this paper is to provide a survey of what we believe are the important recent developments in the area. Our main focus will be on the recent progress made in and the encouraging computational experience gained with the use of optimal solution procedures for the basic resource-constrained project scheduling problem (RCPSP) and important extensions. We illustrate how the branching rules, dominance and bounding arguments of a new depth-first branch-and-bound procedure can be extended to a rich variety of related problems: the generalized resource-constrained project scheduling problem, the resource-constrained project scheduling problem with generalized precedence relations, the preemptive resource-constrained project scheduling problem, the resource availability cost problem, and the resource-constrained project scheduling problem with various time/resource(cost) trade-offs and discounted cash flows.

1 The resource-constrained project scheduling problem (RCPSP)

Resource-constrained project scheduling involves the scheduling of project activities subject to precedence and resource constraints. We assume that a project is represented by an activity-on-the-node network $G = (V,E)$ in which V denotes the set of vertices (nodes) representing the activities and E is the set of edges (arcs) representing the finish-start precedence relationships with zero time lag. The activities are numbered from 1 to n, where the dummy activities 1 and n mark the start and end of the project. The activities are to be performed without preemption. The fixed duration of an activity is denoted by d_i $(1 \leq i \leq n)$, its starting time by s_i $(1 \leq i \leq n)$ and its finishing time by f_i $(1 \leq i \leq n)$. There are K renewable resource types with r_{ik} $(1 \leq i \leq n,\ 1 \leq k \leq K)$ the constant resource requirement of activity i for resource type k and a_k the constant availability of resource type k. Conceptually, the RCPSP can be formulated as follows:

$$\min \ f_n \qquad [1]$$

subject to

$$f_1 = 0 \qquad [2]$$

$$f_j - d_j \geq f_i \qquad (i,j) \in H \qquad [3]$$

$$\sum_{i \in S_t} r_{ik} \leq a_k \qquad t = 1,2,\ldots,f_n \ ; \ k = 1,2,\ldots K \qquad [4]$$

where H denotes the set of pairs of activities indicating precedence constraints and S_t denotes the set of activities in progress in time interval $]t\text{-}1,t] = \{i \mid f_i - d_i < t \leq f_i\}$. Eq. 2 assigns a completion time of 0 to dummy start activity 1. The precedence constraints given by Eq. 3 indicate that activity j can only be started if all predecessor activities i are completed. The resource constraints given in Eq. 4 indicate that for each time period $]t\text{-}1,t]$ and for each resource type k, the renewable resource amounts required by the activities in progress cannot exceed the resource availability. The objective function is given as Eq. 1: the project duration is minimized by minimizing the completion time of the unique dummy ending activity n.

U. Zimmermann et al., *Operations Research Proceedings 1996*

The RCPSP, which as a generalization of the job-shop scheduling problem is NP-hard in the strong sense (Blazewicz et al. 1983), has been extensively studied in the literature. Previous research on optimal procedures basically involves the use of integer programming (Talbot & Patterson 1978) and implicit enumeration (dynamic programming and branch-and-bound). For a comprehensive review we refer the reader to Herroelen & Demeulemeester (1995) and Özdamar & Ulusoy (1995).

The depth-first *DH-procedure*, developed by Demeulemeester & Herroelen (1992), seems to be the fastest exact solution method for solving the RCPSP. Computational experience with the Patterson problem set confirmed the DH-procedure to be, on the average, almost twelve times faster than the best-first procedure developed by Stinson et al. (1978), previously reported to be the most effective and efficient on this problem set. Subsequent research by Kolisch, Sprecher and Drexl (1995) questioned the use of the 110 problem set and led to the development of *ProGen*, a network generator which allows for the generation of RCPSP problem instances which satisfy preset problem parameters. Computational experience on a total of 480 problem instances, generated on the basis of a full factorial design, revealed that the DH-procedure could optimally solve 428 instances within one hour of computation time on a 386SX processor with 15 MHz clockpulse. This inspired a number of authors (Kolisch et al. 1995, Mingozzi et al. 1995, Brucker et al. 1996) to claim that optimal solution procedures such as the DH-procedure cannot solve hard instances to optimality, even with a large amount of computing time. Mingozzi et al. (1995) presented a new 0-1 linear programming formulation that requires an exponential number of variables, corresponding to all feasible subsets of activities that can be simultaneously executed without violating resource or precedence constraints. They present a tree search algorithm *BBLB3* based on this formulation which can solve the 52 hard KSD instances that could not be solved by the DH-procedure, while it is on the average 5 times slower on the Patterson test problems. They conclude that BBLB3 is competitive with the DH-procedure on hard instances, while it does not dominate DH on easier problems. Brucker et al. (1996) developed a branch-and-bound algorithm which performs a depth-first search on a binary search tree, the nodes of which correspond to so-called schedule schemes (sets of disjunctions, conjunctions, parallelity and flexibility relations). The authors develop various bounding and dominance rules and concepts of immediate selection. They report on computational experience with their algorithm on a subset of the Kolisch problems. The algorithm fails to terminate on 8 of the so-called hard instances, while it also fails to terminate on 20 of the so-called easy instances.

Demeulemeester & Herroelen (1995) report, however, that a close look at the 52 problems that could not be solved to optimality by the DH-procedure within the imposed time limit of one hour, indicated that the dominant factor which kept the procedure from finding the optimal solution, was not so much the computation time spent (as could be assumed from the results), but mainly the size of the computer memory that could be addressed. Recent advances in 32 bit-compiler technology inspired the authors to revise and extend the procedure and to implement a new code (subsequently referred to as the *new DH-procedure*) (Demeulemeester & Herroelen 1995). This resulted in a speed boost by a factor of almost three on the 110 Patterson problems as compared to the code used for the 1992 paper, but, more importantly, all the 480 Kolisch, Sprecher and Drexl (KSD) instances could now be solved optimally. Moreover, a truncated version of the procedure yields excellent results. For many KSD instances the first solution found by the new DH-procedure is better than the one found by the popular MINSLK heuristic. Running the new DH-procedure for small amounts of time yields solutions which are very close to the optimum. These results constitute a new benchmark for the RCPSP. Moreover, the efficient and effective branching scheme as well as many of the lower bound and dominance arguments can be extended to a wide and relevant variety of problem settings.

2 The DH- and the new DH-procedure

The *new DH-procedure* (Demeulemeester & Herroelen 1995) is conceptually almost identical to the *DH-procedure* described in Demeulemeester & Herroelen (1992). It generates a search tree, the nodes of which correspond with partial schedules in which finish times temporarily have been assigned to a subset of the activities of the project. The partial schedules are feasible, satisfying both the precedence and resource constraints. *Partial schedules* PS_m are only considered at those time instants m which correspond with the completion time of one or more project activities. The partial schedules are constructed by semi-active timetabling. In other words, each activity is started as soon as it can within the precedence and resource constraints. Partial schedules are built up starting at time 0 and proceed systematically throughout the search process by adding at each decision point subsets of activities, including the empty set, until a complete feasible schedule is obtained. In this sense, a complete schedule is a *continuation of a partial schedule.*

At every time instant m the *eligible set* E_m is defined as the set of activities which are not in the partial schedule and whose predecessor activities have finished. These eligible activities can start at time m if the resource constraints are not violated. Demeulemeester and Herroelen (1992) have proven two theorems which allow the procedure, at decision point m, to decide on which eligible activities must be scheduled by themselves, and which pair of eligible activities must be scheduled concurrently.

Theorem 1. *If at time m the partial schedule* PS_m *has no activity in progress and an eligible activity i cannot be scheduled together with any other unscheduled activity at any time instant* $m' \geq m$ *without violating the precedence and resource constraints, then there exists an optimal continuation of the partial schedule with the eligible activity i put in progress (started) at time m.*

Theorem 2. *If at time m the partial schedule* PS_m *has no activity in progress, if there is an eligible activity i which can be scheduled concurrently with only one other unscheduled activity j at any time instant* $m' \geq m$ *without violating precedence or resource constraints, and if activity j is both eligible and not longer in duration than activity i, then there exists an optimal continuation of the partial schedule in which both activities i and j are put in progress at time m.*

If it is impossible to schedule all activities at time m, a *resource conflict* occurs which will produce a new branching in the branch-and-bound tree. The branches describe ways to resolve the resource conflict by deciding which combinations of activities are to be delayed. A *delaying set D(p)* consists of all subsets of activities D_q, either in progress or eligible, the delay of which would resolve the current resource conflict at level p of the search tree. A *delaying alternative* D_q is minimal if it does not contain other delaying alternatives $D_v \in D(p)$ as a subset. Demeulemeester & Herroelen (1992) give the proof that in order to resolve a resource conflict, it is sufficient to consider only minimal delaying alternatives.

A minimal delaying alternative (node in the search tree) with the smallest lower bound is chosen for branching. The delay of a delaying alternative D_q is accomplished by adding a *temporal constraint* causing the corresponding activities to be delayed up to the *delaying point*, which is defined as the earliest completion of an activity in the set of activities in progress, that does not belong to the delaying alternative. The delayed activities are removed from the partial schedule and the set of activities in progress, and the algorithm continues by computing a new decision point.

The search process continues until the dummy end activity has been scheduled. Every time such a complete schedule has been found, *backtracking* occurs: a new delaying alternative is chosen from the set of delaying alternatives $D(p)$ at the highest level p of the search tree that still has some unexplored delaying alternatives left, and branching continues from that node. When level zero is reached in the search tree, the search process is completed.

Two *dominance rules* are used to prune the search tree. The first one is a variation of the well-known left-shift dominance rule, and can be stated as follows:

Theorem 3. *If the delay of the delaying alternative at the previous level of the branch-and-bound tree forced an activity i to become eligible at time m, if the current decision is to start activity i at time m and if activity i can be left-shifted without violating the precedence or resource constraints (because activities in progress were delayed), then the corresponding partial schedule is dominated.*

The second dominance rule is based on the concept of a cutset. At every time instant m a *cutset* C_m is defined as the set of unscheduled activities for which all predecessor activities belong to the partial schedule PS_m. The proof of the following theorem can be found in Demeulemeester (1992) and Demeulemeester & Herroelen (1992):

Theorem 4. *Consider a cutset C_m at time m which contains the same activities as a cutset C_k, which was previously saved during the search of another path in the search tree. If time k was not greater than time m and if all activities in progress at time k did not finish later than the maximum of m and the finish time of the corresponding activities in PS_m, then the current partial schedule PS_m is dominated.*

The original DH-procedure has been tested with three *lower bounding rules*. The well-known remaining critical path length bound *LB0* and critical sequence lower bound *LB1* (Stinson et al. 1978) are supplemented by an extended critical sequence lower bound *LB2* which is computed by repetitively looking at a path of unscheduled, non-critical activities in combination with a critical path. The *LB2* calculation starts by calculating *LB1*. This allows to determine which activities cannot be scheduled within their slack time. Subsequently, all paths which consist of at least two unscheduled, non-critical activities, and which start and finish with an activity that cannot be scheduled within its slack time, are constructed. A simple type of dynamic programming then allows for the calculation of the extended critical sequence bound for every non-critical path. Subsequent research revealed that *LB0* outperformed the critical sequence lower bounds *LB1* and *LB2*, when used in combination with the cutset dominance pruning rule. As a result, both *LB1* and *LB2* have been removed from the procedure. Moreover, Mingozzi et al. (1995) have introduced a new lower bound *LB3*, based on a new mathematical formulation for the RCPSP and implemented by using a heuristic for solving a set packing problem. Demeulemeester & Herroelen (1995) have incorporated their own version of *LB3* in the new DH-procedure based on the following heuristic. For each activity $i \in A$ they determine its possible *companions,* i.e., the activities with which it can be scheduled in parallel, respecting both the precedence and resource constraints. All unscheduled activities i with a non-zero duration are then entered in a list L in non-decreasing order of the number of companions (non-increasing duration as a tie-breaker). The following procedure then yields a lower bound, *LB3*, for the partial schedule under consideration:

```
LB3:= the earliest completion time of the activities in progress
while list L not empty do
    Take activity j on top of list L and determine its duration $d_j$
    LB3 := LB3 + $d_j$
    Remove activity j and its companions from list L
enddo
```

It is clear that other (more computationally intensive) heuristics can be used to calculate the lower bound *LB3*. The procedure described here is very fast and generally improves the critical path lower bound, *LB0*, if there are pairs of activities that can be scheduled in parallel taking into consideration the precedence constraints only, but cannot be scheduled in this manner if resource constraints are taken into consideration.

The fundamental conclusions which can be drawn from the research on branch-and-bound schemes for the RCPSP can be summarized as follows:

(i) a depth-first branch-and-bound search strategy based on resolving resource conflicts by delaying minimal subsets of activities is a clear favourite for optimally solving RCPSP instances;
(ii) the cutset dominance rule ranks amongst the most effective dominance pruning rules, especially if a sufficient amount of memory (e.g. 24 Mb) can be used for storing the cutsets;
(iii) the use of easy to compute and effective lower bounds (e.g. *LB3* and its possible variations) has a strong impact on the computational cost;
(iv) truncated branch-and-bound procedures provide a suitable alternative for priority based heuristics such as MINSLK;
(v) exploiting the full potential of 32-bit programming provided by recent compilers running on personal computer platforms such as Windows NT® and OS/2® may add considerably to the efficiency of the computer code used;
(vi) reproducable optimal benchmark results are available on the 110 Patterson problems and the 480 KSD problems. In order to avoid computational bias and to guarantee that procedures are validated on a relevant spectrum of problem complexity (the complexity of a problem instance is entwined to the procedure used to solve it), computational experience should be reported on the complete problem sets and should not be limited to selected problem subsets assumed to be "hard" or "easy".

3 Extensions

Various solution concepts used in the DH- and new DH-procedure have been extended to other important problem settings. Demeulemeester & Herroelen (1996b) extend the solution concepts of the DH-procedure to the preemptive resource-constrained project scheduling problem (PRCPSP), which allows for the preemption of activities at integer points in time. Activities are split into subactivities, their number being equal to the duration of the original activity, each having a duration of 1 and resource requirements that are equal to those of the original activity.

Demeulemeester & Herroelen (1996a) have extended the DH-procedure to the generalized resource-constrained project scheduling problem (GRCPSP), which allows for *minimal* start-start, finish-start, start-finish and finish-finish precedence constraints, ready times, due dates and nonconstant resource availabilities.

The resource-constrained project scheduling problem with generalized precedence relations (RCPSP-GPR) allows for minimal and *maximal* time lags which make semi-active timetabling inappropriate. De Reyck & Herroelen (1996) present a new branch-and-bound procedure based on the concepts of minimal delaying alternatives as developed by Demeulemeester & Herroelen (1992) for the RCPSP and adapted by Icmeli and Erengüç (1996) for the RCPSP with discounted cash flows, which replaces the makespan objective by the maximization of the project net present value. Nodes in the search tree represent the initial project network, described by a distance matrix, extended with extra precedence relations to resolve a resource conflict present in the project network of the parent node. Each delaying alternative can now give rise to several delaying modes, i.e. delaying alternatives being delayed by another activity belonging to the resource violating set of activities. Apart from the use of bounding arguments based on extensions of *LB0* and *LB3*, the procedure introduces a new precedence constraint based subset dominance rule. Encouraging results are obtained with a truncated version of the procedure on different sets of problem instances ranging in size from 10 up to 100 activities with a requirement for 1 up to 8 different resource types.

Demeulemeester (1995) studies the resource availability cost problem (RACP) which aims at determining the cheapest resource availability amounts for which a feasible schedule exists that does not violate the project due date. The algorithm relies on the definition of efficient points derived from the solutions obtained by the DH-procedure. The algorithm is validated against Möhring's procedure (Möhring 1984).

The discrete time/resource trade-off problem (DTRTP) assumes that the duration of an activity is a discrete, non-increasing function of the amount of a single renewable resource. The objective is to schedule each activity in one of its possible execution modes in order to minimize the makespan. Demeulemeester et al. (1996c) present two optimal approaches. The mode generation procedure enumerates all possible mode combinations for the network (a mode combination assigns each activity one of its possible modes) and solves for each mode combination the resulting RCPSP using the new DH-procedure. The integrated procedure evaluates at each decision point feasible partial schedules obtained by enumerating all feasible maximal activity-mode combinations. The procedure applies precedence and resource-based bounds in combination with various dominance rules.

The multi-mode project scheduling problem (MMPSP) considers time/resource and resource/resource trade-offs. Activities must be scheduled in one of their possible execution modes subject to renewable and nonrenewable resource constraints. Sprecher et al. (1994) have extended the DH-procedure by fixing the mode of the eligible activities before putting them in progress and by using the concept of minimal delaying alternatives to resolve resource conflicts. Sprecher & Drexl (1996a,b) use an enumeration scheme based on the concept of a precedence tree introduced by Patterson et al. (1990). The procedure uses several preprocessing and dynamic bounding rules. Extensive computational results are obtained on problem sets generated using ProGen (Kolisch et al. 1995). The authors also study the performance of a truncated version of the algorithm.

Table I summarizes the possible extensions.

Table I. Possible extensions of DH-solution concepts

	RCPSP	PRCPSP	GRCPSP	RCPSP-GPR	DTRTP	MMPSP	RCPSP-DC
semi-active timetabling	yes	yes	yes	no	yes	yes	no
min delaying alternatives (or max scheduling alternatives)	yes	yes	yes	yes	yes	yes	yes
delaying alternatives	yes	yes	yes	no	yes	yes	no
delaying modes	no	no	no	yes	no	no	yes
LB0	yes	yes	yes	yes	yes	yes	no
LB1	yes	yes	no	no	yes	yes	no
LB2	yes	yes	no	no	yes	yes	no
LB3	yes	yes	yes	yes	yes	yes	no
Theorem 3	yes	yes	yes	no	yes	yes	no
Theorem 1	yes	yes	yes	no	yes	yes	no
Theorem 2	yes	yes	yes	no	yes	yes	no
Theorem 4	yes	yes	yes	no	yes	yes	no
subset dominance	yes	yes	yes	yes	yes	yes	yes

References

Blazewicz, J., J.K. Lenstra and A.H.G. Rinnooy Kan (1983), Scheduling Projects to Resource Constraints: Classification and Complexity, *Discrete Applied Mathematics*, 5, 11-24.

Brucker, P. A. Schoo and O. Thiele (1996), A Branch-and-Bound Algorithm for the Resource-Constrained Project Scheduling Problem, Preprint Reihe P, Heft 178, Osnabrücker Schriften zur Mathematik, Universität Osnabrück.

Demeulemeester, E. (1992), Optimal Algorithms for Various Classes of Multiple Resource-Constrained Project Scheduling Problems, Ph.D. Thesis, Department of Applied Economic Sciences, Katholieke Universiteit Leuven.

Demeulemeester, E. (1995), Minimizing Resource Availability Costs in Time-Limited Project Networks, *Management Science*, 41, 1590-1598.

Demeulemeester, E. and W. Herroelen (1992), A Branch-and-Bound Procedure for the Multiple Resource-Constrained Project Scheduling Problem, *Management Science*, 38, 1803-1818.

Demeulemeester, E.L. and W.S. Herroelen (1995), New Benchmark Results for the Resource-Constrained Project Scheduling Problem, Research Report N° 9521, Department of Applied Economics, K.U.Leuven.

Demeulemeester, E. and W. Herroelen (1996a), A Branch-and-Bound Procedure for the Generalized Resource-Constrained Project Scheduling Problem, *Operations Research*, to appear.

Demeulemeester, E. and W. Herroelen (1996b), An Efficient Optimal Solution Procedure for the Preemptive Resource-Constrained Project Scheduling Problem, *European Journal of Operational Research*, 90, 334-348.

Demeulemeester, E., B. De Reyck and W. Herroelen (1996c), Optimal and Suboptimal Procedures for the Discrete Time/Resource Trade-Off problem in Project Networks, *PMS'96 - The Fifth International Workshop on Project Management and Scheduling*, April 11-13, Poznan, 84-87.

De Reyck, B. and W. Herroelen (1996), A Branch-and-Bound procedure for the Resource-Constrained Project Scheduling Problem with Generalized Precedence Constraints, Research Report N° 9613, Department of Applied Economics, K.U.Leuven.

Herroelen, W.S. and E.L. Demeulemeester (1995), Recent Advances in Branch-and-Bound Procedures for Resource-Constrained Project Scheduling Problems,Chapter 12 in *Scheduling Theory and Its Applications* (Chrétienne, et al. (eds.)), John Wiley.& Sons.

Icmeli, O. and S. Erenguç (1996), A Branch-and-Bound Procedure for the Resource-Constrained Project Scheduling Problem with Discounted Cash Flows, *Management Science*, to appear.

Kolisch, R., A. Sprecher and A. Drexl (1995), Characterization and Generation of a General Class of Resource-Constrained Project Scheduling problems, *Management Science*, 41, 1693-1703.

Mingozzi, A., V. Maniezzo, S. Ricciardelli and L. Bianco (1995), An Exact Algorithm for Project Scheduling with Resource Constraints Based on a New Mathematical Formulation, Revised Technical Report, University of Bologna, October 1995.

Möhring,, R.H. (1984), Minimizing Costs of Resource Requirements in Project Networks Subject to a Fixed Completion Time, *Operations Research*, 32, 89-120.

Özdamar, L. and G. Ulusoy (1995), A Survey on the Resource-Constrained Project Scheduling Problem, *IIE Transactions*, 27, 574-586.

Patterson, J.H., R. Slowinski, F.B. Talbot and J. Weglarz (1990), Computational Experience with a Backtracking Algorithm for Solving a General Class of Precedence and Resource-Constrained Scheduling problems, *European Journal of Operational Research*, 49, 68-79.

Sprecher, A. and A. Drexl (1996a), Solving Multi-Mode Resource-Constrained Project Scheduling Problems by a Simple, General and Powerful Sequencing Algorithm. Part I: Theory, Research report 385, Christian-Albrechts-Universität zu Kiel, Germany.

Sprecher, A. and A. Drexl (1996b), Solving Multi-Mode Resource-Constrained Project Scheduling Problems by a Simple, General and Powerful Sequencing Algorithm. Part II: Computation, Research report 386, Christian-Albrechts-Universität zu Kiel, Germany.

Sprecher, A., S. Hartmann and A. Drexl (1994), Project Scheduling with Discrete Time-Resource and Resource-Resource Trade-Offs, Research report 357, Christian-Albrechts-Universität zu Kiel, Germany.

Stinson, J.P., E.W. Davis and B.M. Khumawala (1978), Multiple Resource-Constrained Scheduling Using Branch-and-Bound, *AIIE Transactions*, 10, 252-259.

Talbot, B. and J.H. Patterson (1978), An Efficient Integer Programming Algorithm with Network Cuts for Solving Resource-Constrained Scheduling Problems, *Management Science*, 24, 1163-1174.