Manuskripte

aus den

Instituten für Betriebswirtschaftslehre

der Universität Kiel

No. 426

Minimal Delaying Alternatives and Semi-Active Timetabling
in Resource-Constrained Project Scheduling[1]

Arno Sprecher and Andreas Drexl

December 1996

Arno Sprecher, Andreas Drexl, Institut für Betriebswirtschaftslehre, Christian-Albrechts-Universität zu Kiel, Olshausenstraße 40, 24098 Kiel, Germany
Email: sprecher@bwl.uni-kiel.de, drexl@bwl.uni-kiel.de, WWW: http://www.wiso.uni-kiel.de/bwlinstitute/prod, FTP: ftp ://www.wiso.uni-kiel.de/pub/operations-research

[1]Supported by the Deutsche Forschungsgemeinschaft

Abstract

Over the years numerous branch-and-bound procedures for solving the resource-constrained project scheduling problem have been developed. Enumerating delaying alternatives, extension alternatives, feasible posets, feasible sequences or feasible subsets, they all aim at finding as fast as possible a makespan minimal schedule among the resource and precedence feasible ones.

The enumeration is oftenly reduced to the dominant set of semi-active schedules by checking feasibility of local left-shifts. In this paper we show that combining the concepts of minimal delaying alternatives and local left-shifts, if not properly done, does not, as claimed in the literature, reduce the enumeration to the set of semi-active schedules.

Keywords: Project Scheduling, Resource Constraints, Branch-and-Bound, Delaying-Sets, Semi-Active Schedules.

1 Introduction

The development of algorithms for project scheduling has its beginnings in the early fifties when CPM and MPM were formed to support the project manager in doing his work. Given deterministic durations of the activities that built up the project, and precedence relations between some of them, both methods mainly determine time-windows, i.e., intervals, in which the activities can be performed without violating a given project completion time, i.e., makespan. The limitation of the resources required to execute the activities have not been taken into account explicitly.

Since the limitation of the resource availability cannot be relaxed in the major part of business applications the research community has answered the more realistic assumptions by intensive research. As a generalization of the flow-shop, job-shop, and open shop problem, the resource-constrained project scheduling problem (RCPSP) is known as an NP-hard problem. Therefore the main focus is on the development of branch-and-bound algorithms where different ideas have been presented to built the tree guiding the enumeration of the schedules. The schemes proposed enumerate, e.g., delaying alternatives (cf. [4]), extension alternatives (cf.[16]), feasible posets (cf. [11]), feasible sequences (cf. [12],[13]), and feasible subsets (cf. [10])in order to find a schedule with a minimum makespan.

The currently most advanced procedure has been developed by Demeulemeester and Herroelen (cf. [6]). It builds on ideas from Christofides et al. (cf. [2]) and enhances their earlier work (cf. [4]) by a bound introduced by Mingozzi et al. (cf. [10]) and the full exploitation of nowadays available 32-bit architecture of personal computers. The procedure has solved the entire set of benchmark problems generated by ProGen (cf. [9]) for the first time. The projects consist of 32 activities (including two dummy activities) and 4 renewable resources. The CPU-time on a personal computer (80486, 25 MHz,

32 MB) under Windows NT averages at some 34 seconds at the cost of 24 MB used core memory. Unfortunately, in their work Demeulemeester and Herroelen (cf. [4]) claim to perform semi-active timetabling which does not hold. Once planted it served as a spring for numerous publications (cf. [1], [3], [5], [6], [7], and [8]) without correcting the erroneous statement. The attribute, the algorithm does not have, has been used for characterizational purposes.

We will present a simple project instance for which the algorithm proposed by Demeulemeester and Herroelen does not perform semi-active timetabling. More precisely, the optimal schedule determined by their algorithm is not semi-active. We proceed as follows: In Section 2 we give a more detailed description of the problem. Moreover, the dominating sets of semi-active and active schedules are characterized in Section 3. In Section 4 we summarize the algorithm by Demeulemeester and Herroelen, and finally, in Section 5 we discuss the problems one encounters when trying to reduce the enumeration to the set of semi-active schedules. Conclusions are drawn in Section 6.

2 The Model

In this section we will describe the problem in detail (cf., e.g., [2]): We consider a project which consists of n activities each with a deterministic duration of d_i periods, $i = 1, \ldots, n$. Defined by technological requirements a subset H of the cartesian product of the set of activities represents the precedence relations. That is, a given pair $(i, j) \in H$ indicates that activity i has to be completed before activity j is started. Moreover, we assume that the activity-on-node representation of the project has a single source, i.e., the dummy start activity 1, and a single sink, i.e., the dummy finish activity n. The network is acyclic. K resources can be used by the activities. The availability of resource k is b_k units, $k = 1, \ldots, K$, in each period of the processing of the project. Performing an activity i, $i = 1, \ldots, n$, requires r_{ik} units of resource k, $k = 1, \ldots, K$, in each period of its processing time and is not preemptable. The objective is to finish the project as early as possible without violating the precedence and resource constraints. A mathematical programming formulation can be found in [2].

3 Semi-Active Schedules and Local Left-Shifts

For the majority of combinatorial problems branch-and-bound procedures are developed for their solution. Efficient bounds, dominance concepts and characterizations of optimal solutions speed up the convergence of the enumeration scheme. For the resource-constrained project scheduling problem

it is well-known that the set of semi-active and the set of active schedules are dominant sets w.r.t. any regular measure of performance, as, e.g., the minimization of the makespan. We give a verbal description and refer for the technical details as well as for the literature to [15].

By definition a *feasible* schedule assigns each activity of the project a start time, or equivalently a completion time, such that, none of the precedence and resource constraints is violated. Given a feasible schedule, a *left-shift* of an activity reduces the activity's start time without causing a violation of the constraints in the schedule derived. Clearly, not all the left-shifts can be obtained by successively applying the so-called *one-period* left-shifts, that is, a left-shift that reduces the start time by one period. If a left shift is obtained by successive one-period left-shifts of one and the same activity then it is called a *local* left-shift, otherwise it is called a *global* left-shift. Using the distinction of left-shifts we obtain the set of *semi-active* schedules as the ones where no activity can be locally left-shifted and the set of *active* schedules as the ones where no activity can be left-shifted at all, i.e., neither locally nor globally.

As a direct implication of the definitions the set of active schedules is a subset of the set of semi-active schedules. Both sets are dominant with respect to the minimization of the makespan. That is, for minimizing the project's makespan it suffices to examine the semi-active or active schedules. Additionally, by definition, both sets are dominant with respect to any *regular measure of performance*, as, e.g., the minimization of the number of tardy activities.

4 The Search Process

The procedure proposed by Demeulemeester and Herroelen (cf. [4]) that we summarize in the sequel is a depth-first search branch-and-bound approach:

It starts to built the branch-and-bound tree on level $p = 0$. On this level the unique (dummy) start activity, activity 1, is put into progress with a finish time $f_1 = 0$ and the *partial schedule* PS is given by the set $\{1\}$. Note, the decision to start an activity is temporarily made in the sense that it might be delayed on a higher level of the branch-and-bound tree. The *set of activities in progress* S is updated and the next *decision point* m is determined by the minimum finish time of the activities in progress. At this time instant m, the *set of finished activities* F_p, the *set of unfinished activities* U_p, and the *set of eligible activities* E_p are determined. Whereas the set F_p contains all the activities of the partial schedule that have a finish time less than or equal to time instant m, the set U_p is built by the activities out of the partial schedule that are not finished. The set of eligible activities E_p consists of the activities which are not in the partial schedule and whose predecessors are finished. Now, at

time instant m, all the eligible activities are put into progress and the partial schedule as well as the set of activities in progress are updated. If no resource conflict occurs then the next time instant is computed. Otherwise, if a resource conflict occurs at the current time instant, then the level index is incremented and branching is performed on the basis of (minimal) delaying alternatives. For this, a *delaying set* $D(p)$ set is defined as the set of all subsets D_q of activities, either in process or eligible, the delay of which would resolve the resource conflict at level p of the search tree. These subsets are called *delaying alternatives*. A delaying alternative is minimal if it does not contain another delaying alternative as a proper subset. Note, delaying of the activities out of the delaying alternative D_q can be realized by adding a set of extra precedence relations $G_q = \{(j,i)\}$ to the network. G_q is constructed by adding for each activity $i \in D_q$ a precedence relation (j,i) with activity j being an earliest finishing activity, of the ones either in process or eligible to start at time m and that is not delayed (ties are broken arbitrarily). By this concept the partial schedules are successively continued following the objective to find a complete schedule that improves the currently best known makespan. Backtracking is performed, if on a certain level of the branch-and-bound tree there is no (minimal) delaying alternative left to be studied.

The basic scheme, as described above, is restricted to minimal delaying alternatives and enhanced by further dominance and bounding concepts, as, e.g., a variant of the cut-set rule, and the critical sequence lower bound (cf [16]). Moreover, the authors seek to reduce the enumeration to semi-active schedules, described in the previous section, as follows (cf. [4], p. 1805, 1807): Let D_q^* be the delaying alternative currently selected. The set $DS = \{j \in D_q^*; f_j < m + d_j\}$ is defined as the set of activities that have been started earlier than at time instant m, but now have, in accordance with the selection of the minimal delaying alternative D_q^*, to be delayed. "If DS is not empty, the left-shift dominance rule is invoked by using the following selection structure. If the precedence relationships which were added at previous levels of the search tree forced activity i to become eligible at time m, if the current decision was to start that activity at time m and if delaying activity set DS would allow activity i to be left-shifted without causing a resource conflict, then the corresponding partial schedule is dominated" (cf. [4], p. 1807). Or equivalently, if the current decision is to put an activity i in progress which has been delayed on a previous level, i.e., $i \in D_{q-1}^*$, and to delay an activity that has been previously put in progress, then, if activity i can be left-shifted the partial schedule is dominated.

Note, in [4], it is not explicitly stated, if only simple local left-shifts (to the previous decision point) or if the more complex global left-shifts are tested, too. The computational effort differs substantially. However, independently of what is studied feasibility of local left-shifts only, or feasibility of both, local and global left-shifts, the algorithm does not perform semi-active time-tabling as we will see.

5 Counterexample

In this section we will apply the enumeration procedure proposed by Demeulemeester and Herroelen to a project instance. The instance will show that the algorithm, against the authors' assertion, does not only generate semi-active schedules. The instance is displayed in Figure 1.

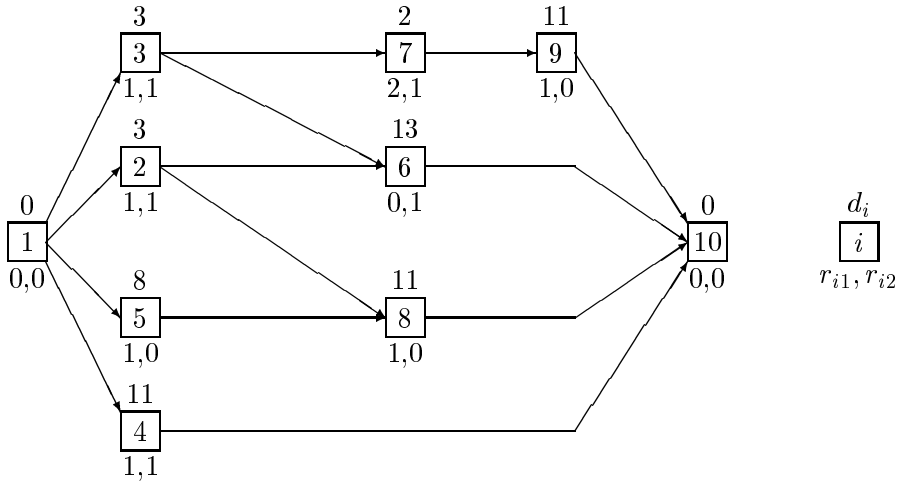

Figure 1: Project Instance

Two renewable resources with an availability of $b_1 = 3$ and $b_2 = 2$ units per period have to be taken into account. The procedure starts by adding the dummy source activity to the partial schedule PS and to the set of activities in progress S, i.e., we obtain $PS = \{1\}$ and $S = \{1\}$. The completion time assigned to activity 1 is $f_1 = 0$. Afterwards the lower bound $LB(0)$ of the project length is determined by the length of the critical path, i.e., 19 periods.

Then, the decision point m is calculated, as the minimum completion time of the activities in process. We obtain $m = 0$ and eliminate those activities from the set of activities in process which finish at the decision point. Subsequently the set of eligible activities E, $E = \{2, 3, 4, 5\}$, is determined. The eligible activities E are put in progress, i.e., $PS = PS \cup E$, and $S = S \cup E$, with finishing times defined by defined by $f_j = m + d_j$. As illustrated in the first Gantt chart of Figure 2, the sum of the resource requests of the activities in process, represented by $[j|r_{j1}, r_{j2}]$, exceeds the availability, and we branch to level $p = 1$. On level 1, the minimal delaying alternatives $D_1 = \{2\}$, $D_2 = \{3\}$, $D_3 = \{4\}$ are determined. They can be realized by the precedence relations $G_1 = \{(3, 2)\}$, $G_2 = \{(2, 3)\}$, $G_3 = \{(2, 4)\}$ which induce critical path bounds (critical sequence bounds) $L_1 = 19$, $L_2 = 19$, $L_3 = \{(2, 4)\}$. We select the delaying alternative with the smallest bound (ties are broken arbitrarily)

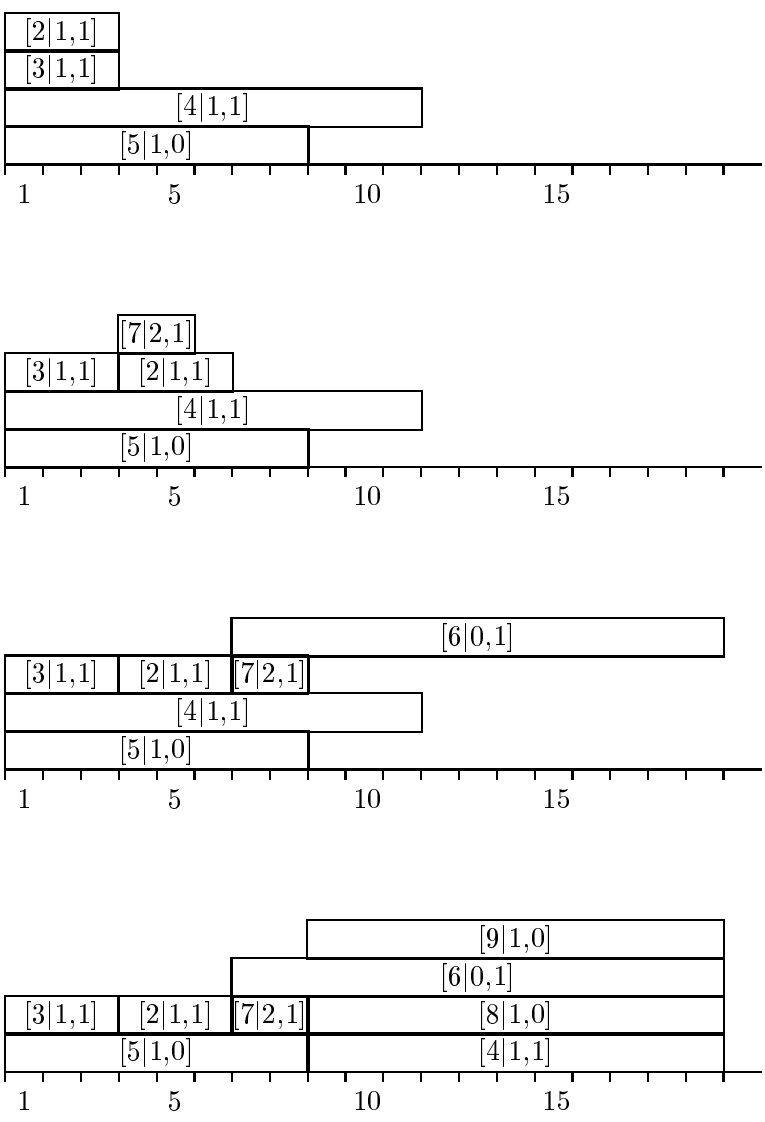

Figure 2: Gantt-Charts of Partial Schedules

$D^* = D_1$ with $G^* = G_1$, and obtain $LB(1) = 19$.

The set of precedence relations H, the partial schedule PS, and the set of activities in process S are updated, i.e., $H = H \cup G^*$, $PS = PS - D^*$, and $S = S - D^*$. None of the activities scheduled at a previous decision point is delayed, i.e., $DS = \emptyset$.

Then the new decision point is determined by the minimum completion time of the activities in process, i.e., $m = f_3 = 3$. After adjusting the set of activities in progress to $S = \{4, 5\}$ and determining the new eligible set $E = \{2, 7\}$, the partial schedule is continued to $PS = PS \cup E = \{1, 2, 3, 4, 5, 7\}$ and the set of activities in progress is updated accordingly, $S = S \cup E = \{2, 4, 5, 7\}$. As to be seen in the second Gantt chart of Figure 2, the sum of the requests of the activities in process at the decision point exceeds the availability, and we branch to a level $p = 2$. On level 2, the minimal delaying

alternatives are $D_1 = \{7\}$, $D_2 = \{2,4\}$, $D_3 = \{2,5\}$, $D_4 = \{4,5\}$, with related precedence sets $G_1 = \{(2,7)\}$, $G_2 = \{(7,2),(7,4)\}$, $G_3 = \{(7,2),(7,5)\}$, $G_4 = \{(7,4)(7,5)\}$, and critical path bounds (critical sequence bounds) $L_1 = 19$, $L_2 = 21$, $L_3 = 24$, $L_4 = 24$. We select $D^* = D_1$ with $G^* = G_1$, and obtain $LB(2) = 19$.

The set of precedence relations H, the partial schedule PS, and the set of activities in process S are updated, i.e., $H = H \cup G^*$, $PS = PS - D^*$, and $S = S - D^*$. None of the activities scheduled at a previous decision point is delayed, i.e., $DS = \emptyset$.

The new decision point is $m = 6$ with active set $S = \{4,5\}$ and eligible set $E = \{6,7\}$. The partial schedule and the set of activities in progress are updated to $PS = PS \cup E = \{1,2,3,4,5,6,7\}$ and $S = S \cup E = \{4,5,6,7\}$ as shown in the third Gantt chart of Figure 2. The sum of the requests of the activities in process at the decision point exceeds the availability, and we branch to level $p = 3$. On level 3, the minimal delaying alternatives are $D_1 = \{4\}$, $D_2 = \{7\}$, $D_3 = \{5,6\}$, with related precedence sets $G_1 = \{(5,4)\}$, $G_2 = \{(5,7)\}$, $G_3 = \{(7,5),(7,6)\}$, and critical path bounds (critical sequence bounds) $L_1 = 19$, $L_2 = 21$, $L_3 = 27$. We select $D^* = D_1$ with $G^* = G_1$, and obtain $LB(3) = 19$.

The set of precedence relations H, the partial schedule PS, and the set of activities in process S are updated, i.e., $H = H \cup G^*$, $PS = PS - D^*$, and $S = S - D^*$. Now, activity 4 which has been scheduled at a previous decision point is delayed, i.e., $DS = \{4\}$. The precedence relation $(2,7)$ forced activity 7 to become eligible at the current decision point $m = 6$, but the delay of activity 4 does not allow activity 7 to be left-shifted, therefore we have to proceed with the determination of the next decision point $m = 8$. However, one can easily see that activity 2 started at $s_2 = 3$ could be left-shifted to start at $\overline{s}_2 = 0$.

Rescheduling of activity 4 and scheduling of activity 8 and 9 at the decision point $m = 8$ leads to the schedule displayed in the fourth Gantt chart. No resource conflict occurs and the schedule is completed by scheduling activity 10. The makespan is 19 periods. Since the makespan coincides with the critical path bound the algorithm terminates. But, since activity 2 can be (locally) left-shifted the schedule derived is not semi-active.

In contrast to the procedures presented by Sprecher and Drexl (cf. [12]) and Stinson et al. (cf. [16]) it is not sufficient to check feasibility of local left-shifts only on the activities put in progress at the current decision point to guarantee that only semi-active schedules are generated. That is, although "conceptual identical" in their application (cf. [4], p. 1807), there is fundamental difference in the effect. To reduce the enumeration to semi-active schedules it is necessary to modify the test: If the current decision is to delay an activity put in process at a previous decision point, i.e., $DS \neq \emptyset$, then

we have to study those activities out of the partial schedule that are assigned a start time ST, $ST > \min\{ST_i; i \in DS\}$. If there is one which can be locally left-shifted then a completion related to the current delaying alternative cannot be semi-active. Moreover, the modification obviously guarantees that only semi-active (partial) schedules are generated.

6 Conclusions

We have studied an algorithm employing the concept of delaying alternatives to find makespan minimal schedules. An instance illustrated that the algorithm does not, as claimed by Demeulemeester and Herroelen, perform semi-active timetabling. The modification proposed can enhance their enumeration scheme to guarantee the construction of semi-active (partial) schedules only. Doing so, the branch-and-bound tree can be reduced substantially.

Acknowledgements: The authors wish to thank Sönke Hartmann for helpful comments and suggestions.

References

[1] Brucker, P.; A. Schoo and O. Thiele (1996): A Branch & Bound algorithm for the resource-constrained project scheduling problem. Research Report, No. 178, Department of Mathematics and Computer Sciences, Osnabrück University, Germany.

[2] Christofides, N.; R. Alvarez-Valdes and J.M. Tamarit (1987): Project scheduling with resource constraints: A branch and bound approach. European Journal of Operational Research, Vol. 29, pp. 262-273.

[3] Demeulemeester, E. (1995): Minimizing resource availability intime-limited projects. Management Science, Vol. 41, pp. 1590–1599.

[4] Demeulemeester, E. L. and W.S. Herroelen (1992): A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. Management Science, Vol. 38, pp. 1803-1818.

[5] Demeulemeester, E. and W. Herroelen (1996a): A Branch-and-Bound algorithm for the generalized resource-constrained project scheduling problem. Operations Research, to appear.

[6] Demeulemeester, E. and W. Herroelen (1996b): New benchmark results for the resource-constrained project scheduling problem. Management Science, to appear.

[7] De Reyck, B. and W. Herroelen (1995): Assembly line balancing by resource-constrained project scheduling techniques – A critical appraisal. Research Report, Department of Applied Economics, University Leuven,Belgium.

[8] Herroelen, W.; E. Demeulemeester and B. De Reyck (1996): Resource-constrained project scheduling: A survey of recent developments. Research Report, Department of Applied Economics, University Leuven,Belgium.

[9] Kolisch, R.; A. Sprecher and A. Drexl (1995): Characterization and generation of a general class of resource-constrained project scheduling problems. Management Science, Vol. 41, pp. 1693-1703.

[10] Mingozzi, A.; V. Maniezzo; S. Ricciardelli and L. Bianco (1996): An exact algorithm for the resource constrained project scheduling problem based on a new mathematical formulation. Management Science, to appear.

[11] Radermacher, F.J. (1985/86): Scheduling of project networks. Annals of Operations Research, Vol. 4, pp. 227-252.

[12] 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. Manuskripte aus den Instituten für Betriebswirtschaftslehre, No. 385, University Kiel, Germany.

[13] 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. Manuskripte aus den Instituten für Betriebswirtschaftslehre, No. 386, University Kiel, Germany.

[14] Sprecher, A.; S. Hartmann and A. Drexl (1996): An exact algorithm for project scheduling with multiple modes. OR Spektrum, to appear.

[15] Sprecher, A.; R. Kolisch and A. Drexl (1995): Semi-active, active and non-delay schedules for the resource-constrained project scheduling problem. European Journal of Operational Research, Vol. 80 , pp. 94-102.

[16] Stinson, J.P.; E.W. Davis and B.M. Khumawala (1978): Multiple resource-constrained scheduling using branch and bound. AIIE Transactions, Vol. 10, pp. 252-259.