Kolisch, Rainer

Efficient priority rules for the resource-constrained project scheduling problem

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 350

Provided in Cooperation with:
Christian-Albrechts-University of Kiel, Institute of Business Administration

Suggested Citation: Kolisch, Rainer (1994) : Efficient priority rules for the resource-constrained project scheduling problem, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 350, Universität Kiel, Institut für Betriebswirtschaftslehre, Kiel

This Version is available at:
https://hdl.handle.net/10419/155422

No. 350

Efficient Priority Rules for the Resource-Constrained Project Scheduling Problem

Rainer Kolisch

August 1994

Rainer Kolisch, Institut für Betriebswirtschaftslehre, Lehrstuhl für Produktion und Logistik,
Christian-Albrechts-Universität zu Kiel, Ohlshausenstr. 40, D-24118 Kiel,
Fax +49-431-880-2072, E-Mail kolisch@bwl.uni-kiel.de

Abstract: We review the well-known RSM priority rule to solve the resource constrained project scheduling problem. A detailed analysis of this rule reveals that it is based on assumptions which do not hold in general and hence might lead to poor solutions. Consequently, an improved RSM priority rule is derived. Furthermore, two new priority rules are developed which extend the well-known precedence based minimum slack priority rule (MSLK) to a precedence and resource based slack priority rule, respectively. An experimental investigation on the basis of two instance-sets from the open literature shows that the new rules outperform all other priority rules which were proposed so far.

Keywords: RESOURCE-CONSTRAINED PROJECT SCHEDULING; PARALLEL SCHEDULING SCHEME; PRIORITY RULES; SINGLE-PASS HEURISTICS; EXPERIMENTAL EVALUATION.

1. Introduction

The classical resource-constrained project scheduling problem (RCPSP) can be stated as follows: We consider a single project which consists of j=1,...,J activities with a non-preemptable duration of d_j periods, respectively. The activities are interrelated by two kinds of constraints: Precedence constraints - as known from traditional CPM-analysis - force an activity not to be started before all its predecessors are finished. Additionally, resource constraints arise as follows: In order to be processed, activity j requires k_{jr} units of resource type $r \in R$ during every period of its duration. Since resource r is only available with the constant period availability of K_r units for each period, activities might not be scheduled at their earliest (precedence feasible) start time but later. The objective of the RCPSP is to schedule the activities such that precedence and resource constraints are obeyed and the makespan of the project is minimised.

Due to its generality the RCPSP is probably one of the most important scheduling problems. Applications of the RCPSP range from project scheduling with the special case CPM to manufacturing where single machine, multiple machines, flow shop, and job shop problems are considered. Furthermore, Leon / Wu (1993) report on applications of the RCPSP to real-time scheduling of automated manufacturing systems and Drexl / Kolisch (1994) show how the RCPSP is embedded into a Leitstand-system.

This paper focuses on the use of priority rules in order to solve the RCPSP in a fast and efficient manner. The remainder is organised as follows: Section 2 gives a formal model of the RCPSP as well as an overview of the available solution procedures. Section 3 presents the parallel scheduling scheme as a framework for priority rules to solve the RCPSP. Based on a classification scheme, classical priority rules are introduced afterwards. One of these rules, the well-known resource scheduling method (RSM) priority rule is analysed in detail. This leads to three new rules which are presented formally and by the use of an example. Section 4 is devoted to an experimental evaluation on the basis of two instance-sets from the open literature. The results show that the new rules outperform the classical priority rules. By employing statistical tests, we verify that the best new rule is significantly better than all other classical rules. Finally, Section 5 is concerned with overall conclusions.

2. Problem Formulation and Solution Procedures

In order to model the RCPSP we make use of the following additional notation: Let P_j define the set of immediate predecessors of activity j. For ease of notation the activities are topologically ordered, i.e. each predecessor of activity j has a smaller number than j. Furthermore, activity j=1 (j=J) is defined to be the unique dummy source (sink) and T denotes an upper bound on the project's makespan. Now, a conceptual model of the RCPSP can be formulated as follows [cf. Talbot / Patterson (1978)]:

$$\text{Min } FT_J \tag{1}$$

subject to

$$FT_i \le FT_j - d_j \qquad j=2,...,J,\ i \in P_j \tag{2}$$

$$\sum_{j \in A_t} k_{jr} \le K_r \qquad r \in R,\ t=1,...,T \tag{3}$$

$$FT_j \ge 0 \qquad j=1,...,J \tag{4}$$

The variable FT_j denotes the (integer valued) finish times of activity j, j=1,...,J, and A_t, the set of activities being in progress in period t, is defined as $A_t := \{ j \mid j=1,...,J,\ FT_j-d_j+1 \le t \le FT_j \}$. The objective function (1) minimises the completion time of the unique sink and thus the makespan of the project. Constraints (2) take into consideration the precedence relations between each pair of activities (i, j), where i immediately precedes j. Finally, constraint set (3) limits the total resource usage within each period to the available amount. Note that (1) to (4) provide no mechanism in order to identify A_t and hence the problem formulation is not amenable to solution via integer programming techniques. To overcome this deficiency, the RCPSP has to be modelled with 0-1 variables as outlined in Pritsker et al. (1969).

The RCPSP is a generalisation of the static job shop problem and hence belongs to the class of *NP*-hard problems [cf. Blazewicz et al. (1983)]. On account of the inherent intractability of the RCPSP, a multitude of exact and heuristic approaches were proposed:

Optimal procedures are *dynamic programming* [cf. Carruthers / Battersby (1966)], *zero-one programming* [cf. Bowman (1959), Pritsker et al. (1969), Patterson / Huber (1974), Patterson / Roth (1976)], as well as implicit enumeration with *branch and bound* [cf. Balas (1971), Davis / Heidorn (1971), Hastings (1972), Radermacher (1985/86), Stinson et al. (1978), Talbot / Patterson (1978), Christofides et al. (1987), Bell / Park (1990), Carlier / Latapie (1991), Demeulemeester / Herroelen (1992)]. Currently, the branch and bound approach of Demeulemeester and Herroelen (1992) seems to be the most powerful optimal procedure available.

With the pioneering works of Kelley (1963) and Brooks [cf. Bedworth / Bailey (1982)] priority rules marked the beginning of **heuristic methods** able to solve the RCPSP. In the following years a multitude of new priority rules were proposed and tested experimentally [cf. Pascoe

(1966), Müller-Merbach (1967), Gonguet (1969), Cooper (1976, 1977), Davies (1973), Patterson (1973), Davis / Patterson (1975), Thesen (1976), Patterson (1976), Whitehouse / Brown (1979), Elsayed (1982), Lawrence (1985), Ulusoy / Özdamar (1989), Alvarez-Tamarit (1989a,b), Boctor (1990), and Valls et al. (1992)]. Recently, the research interests shifted to more elaborate heuristics like truncated branch and bound [cf. Alvarez-Valdes / Tamarit (1989a)], integer programming based heuristics [cf. Oguz / Bala (1994)], disjunctive arc concepts [cf. Shaffer et al. (1965), Alvarez-Valdes / Tamarit (1989a), Bell / Han (1991)], and local search techniques [cf. Sampson / Weiss (1993) and Balakrishnan / Leon (1993)]. With an - in terms of the employed CPU-time - greater effort these heuristics derive superior solutions than simple priority rule procedures.

Nevertheless, for several reasons priority rules are utmost important in order to solve the RCPSP: (i) They are the core of new highly efficient local search based heuristics. E.g., based on the idea of Storer et al. (1992), Balakrishnan / Leon (1993) devised a local search procedure which repeatedly employs a priority rule in order to obtain solutions with perturbed input data. Within sampling heuristics [cf. Levy et al. (1962), Wiest (1967), Cooper (1976), Alvarez-Tamarit (1989b), and Kolisch (1994a)] the priority values of the schedulable activities are biased. That way different solutions can be generated and the solution with the best objective function value is selected. It is intuitive and was experimentally backed up by Kolisch (1994b) that the performance of sampling heuristics depends on the goodness of the priority rule employed. (ii) Commercial project management and scheduling software [e.g., MS-Project, CA-Superproject, etc.] make use of priority rules in order to rapidly derive feasible schedules in the presence of resource constraints. (iii) Finally, priority rules have to be employed when very large projects with more than 1000 activities have to be solved in a fast manner [cf. Valls et al. (1992)]. Hence, it can be summarised that there is a great need for good priority rules.

3. Classical and New Priority Rules for the RCPSP

Generally, a priority rule based scheduling heuristic is made up of two components, a **schedule generation scheme** and a **priority rule**. Two different schemes can be distinguished: The so-called **serial and the parallel scheduling scheme** (where we use the terms scheme and method synonymously). Kolisch (1994b) showed that - when considering a regular measure of performance - for single pass procedures (as studied in this paper) the parallel method is superior to the serial method. Hence, we restrict our focus to the parallel method.

3.1 The Parallel Scheduling Scheme

Today, two algorithms are associated with the so-called **parallel method**: The algorithm of Kelley (1963) and the one of Brooks [cf. Bedworth / Bailey (1982)] which is also termed "Brooks algorithm" (BAG). Like in the majority of publications, the scheduling scheme as proposed by Brooks is employed herein.

The parallel scheduling scheme generates a feasible schedule by extending a partial schedule (i.e., a schedule where only a subset of the activities has been assigned a finish time) in a stage-

wise fashion. There are at most J stages in each of which a set of activities (which might be empty) is scheduled. Associated with each stage n is a schedule time t_n and three disjoint activity-sets: Activities which are completed up to the schedule time are in the *complete set* C_n. Activities which are already scheduled, but during the schedule time still active are in the *active set* A_n. Finally, activities which are available for scheduling w.r.t. precedence and resource constraints but yet unscheduled are in the *decision set* D_n. The partial schedule of each stage is made up by the activities in the complete set and the active set. The schedule time of a stage equals the earliest completion time of activities in the active set of the ancestral stage. Each stage is made up of two steps: (1) The new schedule time is determined and activities with a finish time equal to the (new) schedule time are removed from the active set and put into the complete set. This, in turn, may place a number of activities into the decision set. (2) One activity from the decision set is selected with a priority rule (in case of ties the activity with the smallest label is chosen) and scheduled, respectively started, at the current schedule time. Afterwards, this activity is removed from the decision set and put into the active set. Step (2) is repeated until the decision set is empty, i.e. activities have been scheduled or are not longer available for scheduling w.r.t. resource constraints. The parallel method terminates when all activities are in the complete or active set.

In order to provide a formal description of the parallel scheduling scheme we have to introduce the following notation for πK_r, the left over period capacity of the renewable resource type r at the schedule time t_n, and D_n, the decision set:

$$\pi K_r := K_r - \sum_{j \in A_n} k_{jr}$$

$$D_n := \{ j \mid j \notin \{ C_n \cup A_n \}, P_j \subseteq C_n, k_{jr} \leq \pi K_r \; \forall \, r \in R \}$$

Further, let $v(j)$ be a priority value of activity j, $j \in D_n$. Now, the parallel scheduling scheme can be described as follows:

Initialisation: n:=1, t_n:=0, D_n:={1}, A_n:=C_n:= $\varnothing$, πK_r:=K_r $\forall \, r \in R$, GOTO Step (2);

WHILE $|A_n \cup C_n| < J$ DO **Stage *n***

BEGIN

(1) $t_n := \min \{ FT_j \mid j \in A_{n-1} \}$;
$A_n := A_{n-1} \setminus \{ j \mid j \in A_{n-1}, FT_j = t_n \}$;
$C_n := C_{n-1} \cup \{ j \mid j \in A_{n-1}, FT_j = t_n \}$;
COMPUTE πK_r $\forall \, r \in R$ and D_n;

(2) $j^* := \min_{j \in D_n} \{ j \mid v(j) = \textit{extremum}_{i \in D_n} v(i) \}$;
$FT_{j^*} := t_n + d_{j^*}$;
$A_n := A_n \cup \{ j^* \}$;
COMPUTE πK_r $\forall \, r \in R$ and D_n;
IF $D_n \neq \varnothing$ THEN GOTO Step (2) ELSE n:=n+1;

END;

Stop

3.2 Classical Priority Rules

The first line of step (2) states that for each activity in the decision set a priority value $v(j)$ is computed and the activity with the extremum (i.e. the maximum or minimum) priority value is selected. In case of ties, the activity with the smallest activity number is chosen. The number of priority rules proposed to calculate v is abundant. For the RCPSP, a review can be found in Lawrence (1985) and Alvarez-Valdes / Tamarit (1989a).

Priority rules can be classified according to different criteria. It has to be hastened to say that the classes provided by these criteria are neither exhaustive nor exclusive and are just one way of characterising the abundance of rules presented. (i) A straightforward discrimination w.r.t. the information processed, is to classify priority rules into ***network***, ***time***, and ***resource*** based priority rules [cf. Lawrence (1985) and Alvarez-Valdes / Tamarit (1989a)]. (ii) Another distinction is w.r.t. the dynamic nature of rules: Priority rules which return the same priority value for a certain activity - regardless of the stage they are performed in - are called ***static***, whereas priority rules which may produce different values are called ***dynamic***. (iii) A further discrimination of priority rules is due to the amount of information processed: Rules which employ a small amount of input, usually w.r.t. the activity under consideration only, are regarded to be ***local*** or myopic, while rules which make use of a large amount of information are called ***global***. (iv) Finally, priority rules can be classified into rules where the priority value is a ***lower bound*** or makes use of a lower bound on the one side and rules where no lower bound is part of the priority value on the other side. For classification purposes we use the following notation $\alpha/\beta/\chi/\delta$ with $\alpha=\{N, T, R\}$, $\beta=\{S, D\}$, $\chi=\{L, G\}$, and $\delta=\{B, -\}$. The capital letter uniquely identifies the characteristic via the bold letter of the adjective.

In order to select the best classical priority rules the computational studies of Davis / Patterson (1975), Alvarez-Valdes / Tamarit (1989a), Valls et al. (1992), and Boctor (1990) have been analysed. Table 1 presents the priority rules which ranked among the three best rules in at least one of the studies. The first column contains the name and the abbreviation of the rule, the second column its $\alpha/\beta/\chi/\delta$-classification, the third column indicates if the activity with the maximum or minimum priority value is selected, and the last column gives the formal description of the rule. Additional to the notation already introduced, $\overline{S}_j$ (S_j) denotes the set of all (immediate) successors of activity j and LFT_j (LST_j) denotes the latest finish time (latest start time) of activity j as derived by traditional backward recursion [cf. Elmaghraby (1977)]. Note that for the parallel scheduling scheme the minimum slack priority rule (MSLK) equals the latest start time priority rule (LST) which has been proven by Davis / Patterson (1975). Table 2 summarises the ranking of the three best priority rules in all four publications, respectively, where $a \succ b$ defines that priority rule a has a better performance than rule b.

Priority Rule	Classification	*extr.*	$v(j)$
Most total successors (MTS)	N / S / G / -	*max*	$\lvert S_j \rvert$
Greatest rank positional weight (GRPW)	N,T / S / G / -	*max*	$d_j + \sum_{i \in S_j} d_i$
Latest finish time (LFT)	N,T / S / G / B	*min*	LFT_j
Minimum slack (MSLK)	N,T / D / G / B	*min*	$LST_j - t_n$

Table 1: The Best Classical Priority Rules

Literature	Ranking of Priority Rules
Davis / Patterson (1975)	MSLK $\succ$ LFT $\succ$ RSM
Boctor (1990)	MSLK $\succ$ LFT $\succ$ RSM
Alvarez / Tamarit (1989a)	GRPW $\succ$ LFT $\succ$ MTS
Valls et al. (1992)	MSLK $\succ$ GRPW $\succ$ MTS

Table 2: Ranking of Priority Rules in the Literature

3.3 A Detailed Analysis of the RSM Priority Rule

The only rule listed in Table 2 which is not considered in Table 1 is the so-called RSM priority rule. The fundamental idea of this classical priority rule stems from the disjunctive arc based resource scheduling method (RSM) proposed in Shaffer et al. (1965). The RSM rule was considered in studies undertaken by Davis / Patterson (1975), Alvarez-Valdes / Tamarit (1989a), Boctor (1990), Valls et al. (1992), and Ulusoy / Özdamar (1989). The idea of the RSM rule is as follows: For each pair of activities i and j in the decision set, it is calculated how many periods activity i is delayed beyond its (precedence based) latest start time if it is scheduled after activity j. The activity j which induces the smallest delay of every other activity in the decision set is scheduled. In order to write the RSM rule formally we denote with AP_n the set of all activity pairs in the decision set at stage n, i.e. $AP_n := \{ (i,j) \mid i, j \in D_n, i \neq j \}$. Now, the RSM priority value of activity j, $j \in D_n$, is

$$v(j) = \max \{0, t_n + d_j - LST_i \mid (i,j) \in AP_n \} \tag{5}$$

and the activity with the minimum value is selected. Stated this way, the RSM priority rule implicitly makes use of the following assumption: For each pair of activities i and j, one activity has always to be delayed until the end of the other activity. Let us look at two examples in order to show that this assumption does not hold in general and hence might produce poor results: Consider the project displayed in Figure 1 where we have $|R|=1$ resource type with a period availability of $K_1=4$ units and 6 activities. The associated latest start times derived by backward recursion from the earliest finish time of activity 6, i.e. $LFT_6 = EFT_6 = 5$, are given in Table 3.

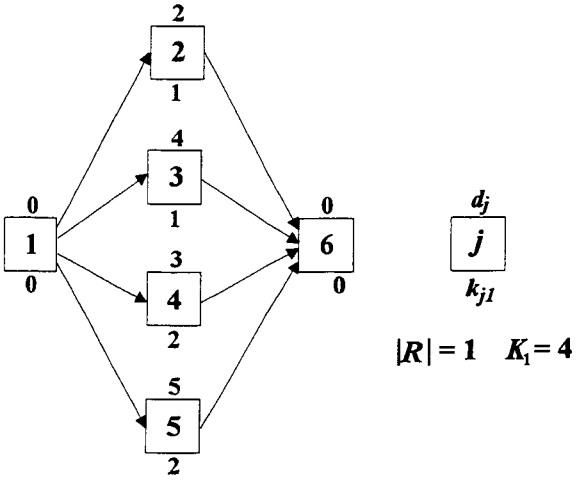

Figure 1: Example Project

j	1	2	3	4	5	6
LST_j	0	3	1	2	0	5

Table 3: Latest Start Times for the Example Project

At stage n=2 the following situation arises: t_n=0, πK_1=4, C_n={1}, A_n=∅, D_n={2,3,4,5}, and AP_n={(2,3),(2,4),(2,5),(3,4),(3,5),(4,5)}. The RSM priority value for activity 2 is calculated to $v(2)$ = max {0, t_2+d_2-LST_3, t_2+d_2-LST_4, t_2+d_2-LST_5} = max {0, 0+2-1, 0+2-2, 0+2-0} = max {0,1,0,2} =2. Presuming that the above stated assumption holds, scheduling activity 2 would induce a delay of two periods for the latest start time of activity 5. Obviously, this is not the case since each of the activity pairs containing activity 2, i.e. (2,3), (2,4), and (2,5), can be started simultaneously at the schedule time.

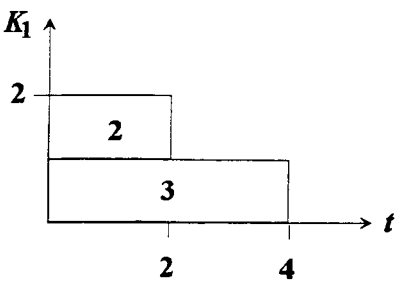

Figure 2: Partial Schedule of the Example Project

Let us now consider that activity 2 and 3 were scheduled and consequently at stage n=2 the partial schedule is as shown by Figure 2, i.e. t_n=0, πK_1=2, C_n={1}, A_n={2,3}, D_n={4,5}, and AP_n={(4,5)}. Actually, at this stage the activity pair (4,5) in the decision set cannot be jointly started. The RSM priority values turn out to be $v(4)$=max {0, t_2+d_4-LST_5} =max {0, 0+3-0} =3 and $v(5)$=max {0, t_2+d_5-LST_4} =max {0, 0+5-2} =3. A look at Figure 3 reveals that scheduling activity 4 indeed forces activity 5 to be delayed until the end of activity 4, i.e. three periods beyond its latest start time as indicated by $v(4)$.

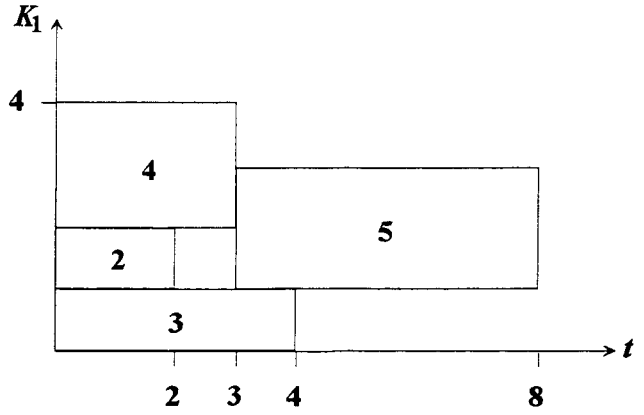

Figure 3: Feasible Solution of the Example Project

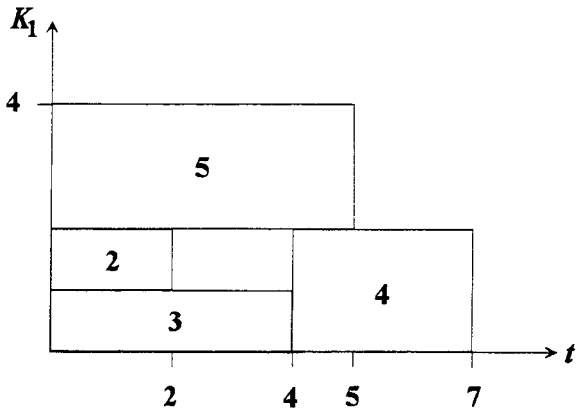

Figure 4: Optimal Solution of the Example Project

But as can be seen in Figure 4, if activity 5 is scheduled, activity 4 is only postponed two periods beyond its latest start time. Hence, the assumption implicitly stated by the RSM priority rule is associated with severe drawbacks. Consequently, the priority rule cannot suggest properly which activity has to be scheduled first.

3.4 Three New Priority Rules for the RCPSP

Let us now refine the RSM priority rule as follows: The set of activity pairs AP_n is divided into three disjoint subsets: The set of generally forbidden pairs (GFP_n) contains activity pairs which - due to resource constraints - can never be scheduled simultaneously, the set of temporarily forbidden pairs (TFP_n) contains pairs of activities which in general can be scheduled simultaneously but due to limited left over capacity cannot be scheduled simultaneously at the schedule time. Finally the set of currently schedulable pairs (CSP_n) contains the activity pairs which can be scheduled simultaneously at the current schedule time. More formally, we denote:

$$GFP_n := \{ (i,j) \mid (i,j) \in AP_n, \exists\, r \in R : k_{ir} + k_{jr} > K_r \} \tag{6}$$

$$TFP_n := \{ (i,j) \mid (i,j) \in AP_n, (i,j) \notin GFP_n, \exists\, r \in R : k_{ir} + k_{jr} > \pi K_r \} \tag{7}$$

$$CSP_n := \{ (i,j) \mid (i,j) \in AP_n, \forall r \in R : k_{ir} + k_{jr} \le \pi K_r \} \tag{8}$$

Since we have three disjoint subsets $AP_n = GFP_n \cup TFP_n \cup CSP_n$ holds. Let us look at our example: At the beginning of stage n=2 we have $GFP_n = TFP_n = \varnothing$ and $CSP_n = AP_n = \{(2,3), (2,4), (2,5), (3,4), (3,5), (4,5)\}$. After activities 2 and 3 were scheduled we have $GFP_n = \varnothing$, $TFP_n = AP_n = \{(4,5)\}$, and $CSP_n = \varnothing$.

Obviously, the earliest time two activities of an activity pair can be started simultaneously is the schedule time t_n if $(i,j) \in CSP_n$ and ∞ if $(i,j) \in GFP_n$. The question is at what period the two activities of a pair can be scheduled simultaneously if they belong to the set TFP_n. Let us consider the following: At stage n, $\Pi^r_{(i,j)}$, the earliest time the temporarily forbidden activity pair (i,j) can be processed simultaneously w.r.t. resource type r only, is given by

$$\Pi^r_{(i,j)} := \min \{ \tau \mid \sum_{h \in A_n \mid FT_h \le \tau} k_{hr} + \pi K_r \ge k_{ir} + k_{jr},\ \tau = t_n, \ldots, T \} \qquad (i,j) \in TFP_n \tag{9}$$

Then, $\Pi_{(i,j)}$, the earliest time to process two activities (i,j) of a temporarily forbidden activity pair simultaneously w.r.t. to all resource types, arises to

$$\Pi_{(i,j)} := \max \{ \Pi^r_{(i,j)} \mid r \in R \} \qquad (i,j) \in TFP_n \tag{10}$$

Now, we can define $\overline{\Pi}_{(i,j)}$, the earliest time for any activity pair (i,j) to be scheduled simultaneously, as

$$\overline{\Pi}_{(i,j)} := \begin{cases} \infty, & \text{if } (i,j) \in GFP \\ \Pi_{(i,j)}, & \text{if } (i,j) \in TFP \\ t_n, & \text{if } (i,j) \in CSP \end{cases} \qquad (i,j) \in AP_n \tag{11}$$

Consequently, for the activity pair (i,j), $E_{(j,i)}$, the earliest time to schedule activity i if activity j is started at the schedule time t_n, is given by

$$E_{(j,i)} := \min \{ t_n + d_j, \overline{\Pi}_{(i,j)} \mid (i,j) \in AP_n \} \tag{12}$$

and $E_{(i,j)}$, the earliest time to schedule activity j if activity i is started at the schedule time t_n, is given by

$$E_{(i,j)} := \min \{ t_n + d_i, \overline{\Pi}_{(i,j)} \mid (i,j) \in AP_n \} \tag{13}$$

Now, an improved RSM priority rule (IRSM) can be stated as follows: Schedule the activity j which induces the smallest increase of the precedence based lower bound for the not chosen activities in the decision set. That is, the IRSM priority value $v(j)$ of activity j, $j \in D_n$, is calculated as

$$v(j) = \max \{0, E_{(j,i)} - LST_i \mid (i,j) \in AP_n \} \tag{14}$$

and the activity with the minimum value is selected. Note that we consider to start activity j, $j \in D_n$, at the current schedule time and hence calculate $E_{(j,i)}$ for all activities i, $i \in D_n$ with $i \neq j$.

Observation 1: If at stage n all activity pairs in the decision set are temporarily schedulable and none of the activities in the decision set has a latest start time which is smaller than the current schedule time, then the IRSM priority value for each activity is "0" and hence the activity in the decision set with the minimum label is selected.

Proof: Since all activity pairs in the decision set are temporarily schedulable, on account of equation (8) we have $AP_n=CSP_n$. Hence, equation (11) gives

$$\overline{\Pi}_{(i,j)} = t_n \qquad (i,j) \in AP_n \qquad (15)$$

Since $d_j \geq 0$ holds for all j, (12) results in

$$E_{(j,i)} = t_n \qquad (i,j) \in AP_n \qquad (16)$$

Furthermore, none of the activities in the decision set has a latest start time which is smaller than the current schedule time, i.e.

$$LST_j \geq t_n \qquad j \in D_n \qquad (17)$$

Substituting (16) into (14) yields

$$v(j) = \max \{0,\ t_n - LST_i \mid (i,j) \in AP_n \} \qquad (18)$$

and on account of (17) we have $v(j) = 0, j \in D_n$.■

Utilising $E_{(i,j)}$ and combining it with the notion of slack another new priority rule can be formulated as follows: Schedule the activity j which - if not chosen - has the worst case slack (WCS). More formally, the WCS priority value of activity $j, j \in D_n$, is calculated as

$$v(j) = LST_j - \max \{ E_{(i,j)} \mid (i,j) \in AP_n \} \qquad (19)$$

and the activity with the minimum value is selected.

Observation 2: If at stage n all activity pairs in the decision set are temporarily schedulable, i.e. resources do not impose constraints, then WCS reduces to MSLK, the best of the classical priority rules (cf. Table 2).

Proof: Since all activity pairs in the decision set are temporarily schedulable, on account of equation (8) we have $AP_n=CSP_n$. Hence, equation (11) once more gives

$$\overline{\Pi}_{(i,j)} = t_n \qquad (i,j) \in AP_n \qquad (20)$$

Since $d_i \geq 0$ holds for all i, (13) results in

$$E_{(i,j)} = t_n \qquad (i,j) \in AP_n \qquad (21)$$

Substituting (21) into (19) yields

$$v(j) = LST_j - t_n \qquad (22)$$

which equals the calculation of the MSLK priority value (cf. Table 1).■

Employing the idea of Fadlalla et al. (1994) for the single machine problem with average tardiness performance measure we can propose the average case slack priority rule (ACS). Here, the activity is selected which - if not chosen - has the largest average case slack value. Precisely, the ACS priority value of activity $j, j \in D_n$, is calculated as

$$v(j) = LST_j - \frac{1}{|D_n|-1} \sum_{(i,j)\in AP_n} E_{(i,j)} \tag{23}$$

and the activity with the minimum value is scheduled.

Observation 3: If at stage n all activity pairs in the decision set are temporarily schedulable then ACS reduces to MSLK too.

Proof: Substituting (21) into (23) yields

$$v(j) = LST_j - t_n \tag{24}$$

which equals the calculation of the MSLK priority value (cf Table 1).■

The three new priority rules are demonstrated by applying them to the example project of this section. Consider stage n=2 of the partial schedule depicted in Figure 2 with t_n=0, πK_1=2, A_n={2,3}, D_n={4,5}. The set of the temporarily forbidden pairs arises to $TFP_n=AP_n$={(4,5)}, i.e., activities 4 and 5 cannot be jointly started at the current schedule time t_n. Consequently, the set of general forbidden pairs and the set of currently schedulable pairs are empty, i.e., $GFP_n=CSP_n=\varnothing$. Now, the earliest time to process activities 4 and 5 simultaneously is $\overline{\Pi}_{45}=\overline{\Pi}_{54}=4$. Then, the earliest time to start activity 5, if activity 4 is scheduled at t_2, is $E_{(4,5)}=\min\{t_2+d_4, \overline{\Pi}_{54}\}=\min\{0+3,4\}=3$. Correspondingly, the earliest time to start activity 4 if activity 5 is scheduled at t_2 is $E_{(5,4)}=\min\{t_2+d_5, \overline{\Pi}_{45}\}=\min\{0+5,4\}=4$.

Hence, the values for the IRSM rule arise to $v(4)=\max\{0,E_{(4,5)}-LST_5\}=\max\{0,3-0\}=3$ and $v(5)=\max\{0, 4-2\}=2$, respectively. I.e., scheduling activity 4 (5) increases the precedence based lower makespan bound by 3 (2) periods. The IRSM rule chooses activity 5 to be started, which leads to the optimal solution as depicted in Figure 4.

Applying the WCS priority rule, the following priority values are calculated: $v(4)=LST_4-\max\{E_{(5,4)}\}=2-4=-2$ and $v(5)=0-3=-3$. That is, if activity 4 (5) is not scheduled at t_2 it has a negative slack of 2 (3) periods. Hence, the WCS priority rule also chooses activity 5 and thus derives the optimal solution.

Finally, since we have $|D_n|$=2 for the above example the ACS priority values are calculated like the WCS priority values.

Table 4 provides an overview of the classical RSM rule and the three new priority rules presented in this section.

Priority Rule	*extr.*	$v(j)$	Classification
Resource Scheduling Method (RSM)	min	$\max \{0, t_n + d_j - LST_i \mid (i,j) \in AP_n \}$	N,T / D / G / -
Impoved RSM (IRSM)	min	$\max \{0, E_{(j,i)} - LST_i \mid (i,j) \in AP_n \}$	N,T,R/ D / G / B
Worst Case Slack (WCS)	min	$LST_j - \max \{ E_{(i,j)} \mid (i,j) \in AP_n \}$	N,T,R/ D / G / B
Average Case Slack (ACS)	min	$LST_j - \frac{1}{\lvert D_n \rvert - 1} \sum_{(i,j) \in AP_n} E_{(i,j)}$	N,T,R/ D / G / B

Table 4: RSM-based Priority Rules

4. Computational Results

In order to evaluate the classical and the new priority rules we employed two instance sets from the open literature. The first one is the well-known set of Patterson (1984) which was originally used to compare four exact solution procedures for the RCPSP. A detailed description of this instance set can be found in Patterson (1984) and Demeulemeester / Herroelen (1992). Briefly, there are 110 problems, each with 7 to 50 activities. An activity requires between one and three resource types. For 103 problems each activity uses all three resource types. Since the Patterson-instances are widely known and accepted as benchmark problems they are employed in this study. However, there are two drawbacks associated with the instance set: Assembled from different sources the problems are not generated by using a controlled design. Furthermore, recent advances in the development of exact procedures [cf. Demelemeester / Herroelen (1992)] showed that the instances can be solved in an average CPU-time of less than 1 second on a personal computer.

In order to have an instance set without the outlined deficiencies we also employed the 308 resource-constrained problems of Kolisch et al. (1992). These instances stem from the 480 problems which were originally generated with the parameter driven problem generator ProGen. Here, we neither considered the 120 resource-unconstrained problems nor the 52 problems for which Kolisch et al. (1992) could not verify the optimal solution. A detailed description of the instances can be found in Kolisch et al. (1992). Briefly, each problem consists of 32 activities and 4 resource types. Each activity requires between 1 and 4 resource types. Like the Patterson-instances, all 308 ProGen-problems were solved with the exact procedure of Demeulemeester / Herroelen (1992) to optimality. We implemented their C-code on a personal computer with 8036sx processor, mathematical coprocessor, and 15 MHz clockpulse. The average solution time was 952,13 CPU-seconds and the maximum time did not exceed 3600 CPU-seconds.

The parallel scheduling scheme and the 5 classical as well as 3 new priority rules have been coded in PASCAL and implemented on an IBM compatible personal computer with 80386dx processor and 40 MHz clockpulse at the laboratory of the Christian-Albrechts-Universität zu Kiel.

4.1 Results for the ProGen-Instances

Table 5 shows the mean (MEAN), the standard deviation (STDDEV), and the maximum (MAX) percentage deviation from the optimal objective function value for each of the 8 priority rules, respectively. Furthermore, the number of optimal solutions (#OPT) and the average CPU-time in seconds (CPU) is denoted. The priority rules are listed w.r.t. to ascending MEAN. It can be seen that the three new priority rules WCS, ACS, and IRSM show better results than all other tested rules. Note that the performance and the ranking of the classical rules obtained by our study is comparable to the results of other studies [cf. Davis / Patterson (1975) and Alvarez-Valdes / Tamarit (1989a)].

	MEAN	STDDEV	MAX	#OPT	CPU
WCS	4,27	4,60	26,39	96	,02
ACS	4,57	4,94	32,81	94	,02
IRSM	4,66	4,83	27,03	95	,02
LFT	4,83	4,77	22,37	85	,01
MSLK	4,92	5,25	32,91	92	,01
RSM	5,67	5,05	22,22	72	,02
MTS	5,92	5,25	32,81	69	,02
GRPW	9,22	8,17	41,86	58	,01

Table 5: Performance of Priority Rules for the ProGen-Instances

	ACS	IRSM	LFT	MSLK	RSM	MTS	GRPW
			δ = B			δ = -	
WCS	,179	,0409	,0006	,0001	,0000	,0000	,0000
ACS		,5069	,0935	,0046	,0000	,0000	,0000
IRSM			,3513	,2790	,0000	,0000	,0000
LFT				,9020	,0000	,0000	,0000
MSLK					,0005	,0000	,0000
RSM						,5451	,0000
MTS							,0000

Table 6: 2-tailed Wilcoxon Matched-Pairs Signed-Ranks Test for the ProGen-Instances

In order to test the significance of the results, we performed the 2-tailed Wilcoxon matched-pairs signed-ranks test for each pair of priority rules (we did not apply the t-test since for none of the priority rules the percentage deviation from the optimal solution was normally distributed). Table 6 shows the confidence levels for each pair of priority rules. Hatched fields signal that there is a significant difference between two rules at the 1%-level of confidence, i.e. $p \leq 0.01$. It can be stated that the WCS-rule significantly outperforms all classical priority

rules. Also, all new RSM-based rules are significantly better than the original RSM priority rule. Another interesting finding is that there are two disjoint classes: Lower bound based rules, i.e. WCS, ACS, IRSM, LFT, and MSLK, and priority rules which do not make use of a lower bound. Each rule of the first class significantly outperforms every rule in the other class.

4.2 Results for the Patterson-Instances

Table 7 and 8 present the results for the Patterson-instances. With one exception (LFT and MSLK) the ranking of the priority rules is identical to the one derived for the ProGen-instances. Applying the Wilcoxon test to the Patterson-instances the discrimination between lower bound based rules and other rules was not as distinctive as for the ProGen-instances. All other results were comparable to the ones already obtained with the ProGen-set.

	MEAN	STDDEV	MAX	#OPT	CPU
WCS	3,71	4,13	16,67	42	,02
ACS	4,12	4,44	22,22	37	,02
IRSM	4,53	4,76	19,51	33	,01
MSLK	5,0	5,68	23,68	34	,01
LFT	5,68	5,25	21,95	30	,01
RSM	6,24	5,3	21,05	22	,01
MTS	6,65	4,8	21,05	20	,02
GRPW	9,47	7,96	42,11	22	,01

Table 7: Performance of Priority Rules for the Patterson-Instances

	ACS	IRSM	MSLK	LFT	RSM	MTS	GRPW
WCS	,0575	,0714	,0012	,0000	,0000	,0000	,0000
ACS		,3156	,0280	,0010	,0000	,0000	,0000
IRSM			,5789	,0096	,0000	,0000	,0000
MSLK				,1543	,0090	,0036	,0000
LFT					,2514	,0278	,0000
RSM						,4255	,0000
MTS							,0002

Table 8: 2-tailed Wilcoxon Matched-Pairs Signed-Ranks Test for the Patterson-Instances

Table 9 gives a direct comparison of the MEAN for the two instance-sets. For each priority rule the hatched field indicates the instance set on which a better performance was obtained. From the results of optimal procedures [cf. Demeulemeester / Herroelen (1992)] it is expected that the performance is better for the Patterson-instances. But this holds only for the new

priority rules while the classical priority rules perform better on the ProGen-instances. Hence, the new priority rules are even more superior when considering the Patterson-instances.

	WCS	ACS	IRSM	LFT	MSLK	RSM	MTS	GRPW	MEAN
ProGen	4,27	4,57	4,66	4,83	4,92	5,67	5,92	9,22	5,50
Patterson	3,71	4,12	4,53	5,68	5,0	6,24	6,65	9,47	5,67

Table 9: Comparison of the Results for the ProGen- and the Patterson-Instances

4.3 Effect of Problem Parameters on the Performance of Priority Rules

In the last two subsections it has been shown that the new priority rules outperform the classical priority rules. An interesting question is if this holds for all types of instances or only for a certain class of problems. For classical rules Kolisch (1994a,b) found out that the ranking of priority rules is not significantly altered by different problem classes. In order to obtain insight for the new priority rules we have once more employed the ProGen-instances. As oulined in the beginning of Section 4, the problems were generated by a controlled design of specified problem parameters. Three problem parameters were employed [for details cf. Kolisch et al. (1992)]: (i) The network complexity (*NC*) is the ratio of non-redundant precedence relations to the number of activities. (ii) The resource factor (*RF*) reflects the density of the two dimensional array k_{jr}, j=2,...,J-1 and r=1,...,$|R|$. That is, for *RF*=1 each non-dummy activity requests the full complement of the $|R|$ resource types while for *RF*=0 none of the activities requests any of the resource types. (iii) Finally, the resource strength (*RS*) measures the degree of resource-constrainedness in the interval [0,1]. For each resource type r, $r \in R$, the resource strength is computed as follows: $RS=(K_r-K_r^{min})/(K_r^{max}-K_r^{min})$, where K_r^{min} is the minimal availability of resource type r in order to assure feasibility of the RCPSP, i.e. $K_r^{min} = \max\{k_{jr} | j=1,...,J\}$, and K_r^{max} is the peak demand of resource type r in a CPM-schedule. That is, for *RS*=1 we have the resource-unconstrained CPM-case, while for *RS*=0 we have a highly resource constrained problem. The following levels of the problem parameters were chosen: *NC* = {1,5; 1,8; 2,1}, *RF* = {0,25; 0,5; 0,75; 1}, and *RS* = {0,2; 0,5; 0,7}.

Figures 5 to 7 show the single-factor effects of the problem parameters on MEAN. It can be seen that there are no major alternations within the ranking of the priority rules. Furthermore, the new priority rules show for all problem parameter settings the best results. Especially WCS does always perform best regardless of the adjustment of the problem parameters. Hence, we can conclude that the new priority rules and especially WCS are highly suitable to solve all types of RCPSP instances.

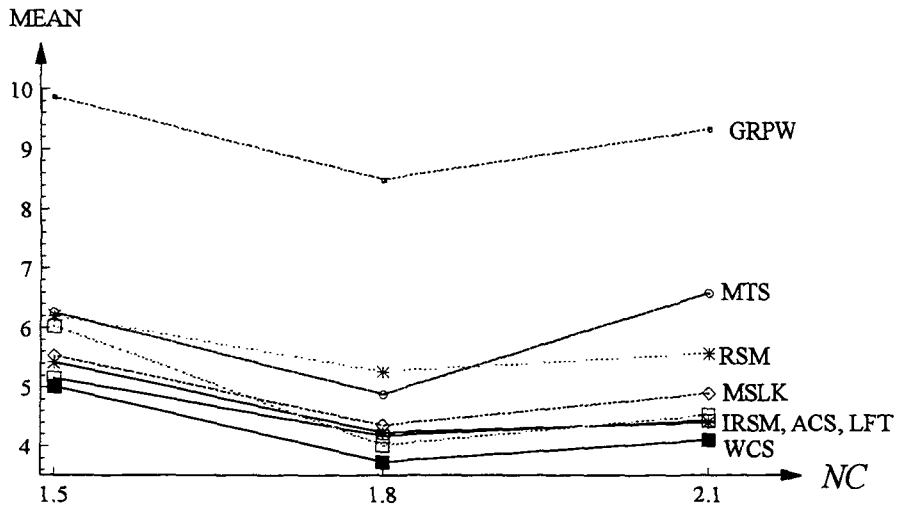

Figure 5: Effect of the Network Complexity (*NC*) on the Performance of Priority Rules

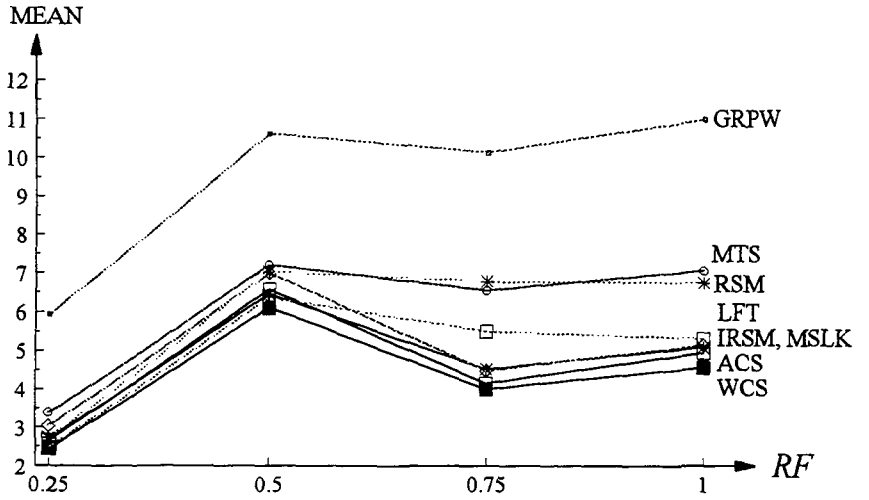

Figure 6: Effect of the Resource Factor (*RF*) on the Performance of Priority Rules

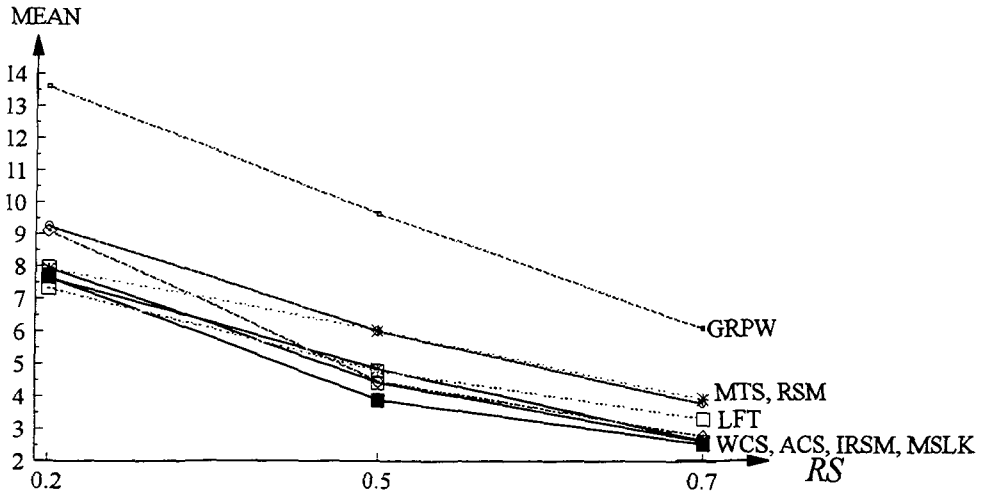

Figure 7: Effect of the Resource Strength (*RS*) on the Performance of Priority Rules

5. Conclusions

In this paper we have reviewed the best classical priority rules available in the open literature for scheduling a project to minimise its total duration subject to technological precedence constraints and resource constraints. A detailed analysis of the well-known RSM priority rule was the stepping stone for the development of three new priority rules. By a thorough computational study with two instance-sets from the literature it has been shown that the new rules outperform all other rules proposed so far. Even more, the best one of the new rules, the so-called WCS priority rule, is on both instance-sets significantly superior to all other priority rules. Hence, this priority rule should serve as the backbone for sophisticated local search heuristics in order to solve the resource constrained project scheduling problem of all problem sizes close to optimality.

Acknowledgement: The author is gratefully indebted to Andreas Drexl, Universität Kiel, James H. Patterson, Indiana University, and to Erik Demeulemeester, Katholieke Universiteit Leuven, respectively, for their help in this research.

References

Alvarez-Valdes, R. and J.M. Tamarit (1989a): Heuristic algorithms for resource-constrained project scheduling: A review and an empirical analysis, in: Slowinski, R. and J. Weglarz (Eds.): Advances in project scheduling, Elsevier, Amsterdam, pp. 113-134.

Alvarez-Valdes, R. and J.M. Tamarit (1989b): Algoritmos heuristicos deterministas y aleatorios en secuenciacion de proyectos con recursos limitados, Qüestiio, Vol. 13, pp. 173-191.

Balakrishnan, R. and V. J. Leon (1993): Quality and adaptability of problem-space based neighborhoods for resource constrained scheduling, Working Paper, Department of Industrial Engineering, Texas A&M University, USA.

Balas, E. (1971): Project scheduling with resource constraints, in: Beale, E.M.L. (Ed.): Applications of mathematical programming techniques, English University Press, London, pp. 187-200.

Bedworth, D.D. and J.E. Bailey (1982): Integrated production control systems - Management, analysis, design, Wiley, New York.

Bell, C.E. and J. Han (1991): A new heuristic solution method in resource-constrained project scheduling, Naval Research Logistics, Vol. 38, pp. 315-331.

Bell, C.E. and K. Park (1990): Solving resource-constrained project scheduling problems by A* search, Naval Research Logistics, Vol. 37, pp. 61-84.

Blazewicz, J., J.K. Lenstra, and A.H.G. Rinnooy Kan (1983): Scheduling subject to resource constraints: Classification and complexity, Discrete Applied Mathematics, Vol. 5, pp. 11-24.

Boctor, F.F. (1990): Some efficient multi-heuristic procedures for resource-constrained project scheduling, European Journal of Operational Research, Vol. 49, pp. 3-13.

Bowman, E.H. (1959): The schedule-sequencing problem, Operations Research, Vol. 7, pp. 621-624.

Carlier, J. and B. Latapie (1991): Une methode arborescente pour resoudre les problemes cumulatifs, Recherche operationnelle, Vol. 25, pp. 311-340.

Carruthers, J.A. and A. Battersby (1966): Advances in critical path methods, Operational Research Quarterly, Vol. 17, pp. 359-380.

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.

Cooper, D.F. (1976): Heuristics for scheduling resource-constrained projects: An experimental investigation, Management Science, Vol. 22, pp. 1186-1194.

Cooper, D.F. (1977): A note on serial and parallel heuristics for resource-constrained project scheduling, Foundations of Control Engineering, Vol. 2, pp. 131-134.

Davies, E.M. (1973): An experimental investigation of resource allocation in multiactivity projects, Operational Research Quarterly, Vol. 24, pp. 587-591.

Davis, E.W. (1966): Resource allocation in project network models - A survey, The Journal of Industrial Engineering, Vol. 17, pp. 177-188.

Davis, E.W. and G.E. Heidorn (1971): An algorithm for optimal project scheduling under multiple resource constraints, Management Science, Vol. 17, pp. 803-816.

Davis, E.W. and J.H. Patterson (1975): A comparison of heuristic and optimum solutions in resource-constrained project scheduling, Management Science, Vol. 21, pp. 944-955.

Demeulemeester, E. 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.

Drexl, A. and R. Kolisch (1994): Model-based assembly management in machine tool manufacturing, Research Report No. 346, Institut für Betriebswirtschaftslehre, Christian-Albrechts-Universität zu Kiel.

Elmaghraby, S.E. (1977): Activity networks: Project planning and control by network models, Wiley, New York.

Elsayed, E.A. (1982): Algorithms for project scheduling with resource constraints, International Journal of Production Research, Vol. 20, pp. 95-103.

Fadlalla, A., J.R. Evans, and M.S. Levy (1994): A greedy heuristic for the mean tardiness sequencing problem, Computers & Operations Research, Vol. 21, pp. 329-336.

Gonguet, L. (1969): Comparison of three heuristic procedures for allocating resources and producing schedules, in: Lombaers, H.J.M. (Ed.): Project planning by network analysis, North-Holland, Amsterdam, pp. 249-255.

Hastings, N.A.J. (1972): On resource allocation in project networks, Operational Research Quarterly, Vol. 23, pp. 217-221.

Kelley, J.E., Jr. (1963): The critical-path method: Resources planning and scheduling, in: Muth, J.F. and G.L. Thompson (Eds.): Industrial scheduling, Prentice-Hall, New Jersey, pp. 347-365.

Kolisch, R. (1994a): Project scheduling under resource constraints - Efficient heuristics for several problem classes, PhD Dissertation, Kiel.

Kolisch, R. (1994b): Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation, Research Report No. 344, Institut für Betriebswirtschaftslehre, Christian-Albrechts-Universität zu Kiel.

Kolisch, R., A. Sprecher, and A. Drexl (1992): Characterization and generation of a general class of resource-constrained project scheduling problems: Easy and hard instances, Research Report No. 301, Institut für Betriebswirtschaftslehre, Christian-Albrechts-Universität zu Kiel (under review for Management Science).

Lawrence, S.R. (1985): Resource-constrained project scheduling - A computational comparison of heuristic scheduling techniques, Working Paper, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, USA.

Leon, V.J. and S.D. Wu (1993): Characterisitcs of computerized scheduling and control of manufacturing systems, in Joshi, S.B. and J.S. Smith (Eds.): Computer control of manufacturing systems, Chapman & Hall, London.

Levy, F.K., G.L. Thompson, and J.D. Wiest (1962): Multiship, multishop, workload-smoothing program, Naval Research Logistics Quarterly, Vol. 9, pp. 37-44.

Müller-Merbach, H. (1967): Ein Verfahren zur Planung des optimalen Betriebsmitteleinsatzes bei der Terminierung von Großprojekten, Zeitschrift für wirtschaftliche Fertigung, Vol. 62, pp. 83-88, 135-140.

Oguz, O. and H. Bala (1994): A comparative study of computational procedures for the resource constrained project scheduling problem, European Journal of Operational Research, Vol. 72, pp. 406-416.

Pascoe, T.L. (1966): Allocation of resources C.P.M., Revue Francaise Recherche Operationelle, No. 38, pp. 31-38.

Patterson, J.H. (1973): Alternate methods of project scheduling with limited resources, Naval Research Logistics Quarterly, Vol. 20, pp. 767-784.

Patterson, J.H. (1976): Project scheduling: The effects of problem structure on heuristic performance, Naval Research Logistics Quarterly, Vol. 23, pp. 95-123.

Patterson, J.H. (1984): A comparison of exact approaches for solving the multiple constrained resource, project scheduling problem, Management Science, Vol. 30, pp. 854-867.

Patterson, J.H. and W.D. Huber (1974): A horizon-varying, zero-one approach to project scheduling, Management Science, Vol. 20, pp. 990-998.

Patterson, J.H. and G.W. Roth (1976): Scheduling a project under multiple resource constraints: A zero-one programming approach, AIIE Transactions, Vol. 8, pp. 449-455.

Pritsker, A.A.B., L.J. Watters, and P.M. Wolfe (1969): Multiproject scheduling with limited resources: A zero-one programming approach, Management Science, Vol. 16, pp. 93-107.

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

Sampson, S.E. and E.N. Weiss (1993): Local search techniques for the generalized resource constrained project scheduling problem, Naval Research Logistics, Vol. 40, pp. 365-375.

Shaffer, L.R., J.B. Ritter, and W.L. Meyer (1965): The critical-path method, McGraw-Hill, New York.

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.

Storer, R.H., S.D. Wu, and R. Vaccari (1992): New search spaces for sequencing problems with application to job shop scheduling, Management Science, Vol. 38, pp. 1495-1509.

Talbot, B. and J.H. Patterson (1978): An efficient integer programming algorithm with network cuts for solving resource-constrained scheduling problems, Management Science, Vol. 24, pp. 1163-1174.

Thesen, A. (1976): Heuristic scheduling of activites under resource and precedence restrictions, Management Science, Vol. 23, pp. 412-422.

Ulusoy, G. and L. Özdamar (1989): Heuristic performance and network / resource characteristics in resource-constrained project scheduling, Journal of the Operational Research Society, Vol. 40, pp. 1145-1152.

Valls, V., M.A. Perez, and M.S. Quintanilla (1992): Heuristic performance in large resource-constrained projects, Working Paper, Departament D'Estadistica I Investigacio Operativa, Universitat De Valencia, Spain.

Whitehouse, G.E. and J.R. Brown (1979): Genres: An extension of Brooks algorithm for project scheduling with resource constraints, Computers & Industrial Engineering, Vol. 3, pp. 261-268.

Wiest, J.D. (1967): A heuristic model for scheduling large projects with limited resources, Management Science, Vol. 13, pp. B359-B377.