ELSEVIER European Journal of Operational Research 90 (1996) 320–333

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation

Rainer Kolisch [*]

Institut für Betriebswirtschaftslehre, Lehrstuhl für Produktion und Logistik, Christian-Albrechts-Universität zu Kiel, Olshausenstr. 40, D-24098 Kiel, Germany

Received August 1994; revised April 1995

Abstract

We consider the so-called parallel and serial scheduling method for the classical resource-constrained project scheduling problem. Theoretical results on the class of schedules generated by each method are provided. Furthermore, an in-depth computational study is undertaken to investigate the relationship of single-pass scheduling and sampling for both methods. It is shown that the performance-ranking of priority rules does not differ for single-pass scheduling and sampling, that sampling improves the performance of single-pass scheduling significantly, and that the parallel method cannot be generally considered as superior.

Keywords: Resource-constrained project scheduling; Serial and parallel scheduling method; Single-pass scheduling and sampling; Active and non-delay schedules; Priority rules; Experimental investigation

1. Introduction

The classical resource-constrained project scheduling problem (RCPSP) can be stated as follows: We consider a single project which consists of $j = 1, \dots, J$ 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 have been finished. Additionally, resource constraints arise as follows: In order to be processed, activity j requires k_{jr} units of resource $r \in R$ 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.

Two of the oldest and best known heuristics for the RCPSP are the serial and the parallel scheduling scheme, respectively (cf. the literature cited in Section 3). Both can be applied as a deterministic single-pass or a probabilistic multi-pass (sampling) method.

The majority of publications dealing with scheduling schemes for the RCPSP report on the perfor-

[*] Fax +49-431-880-2072, E-mail: kolisch@bwl.uni-kiel.de

0377-2217/95/$09.50 © 1995 Elsevier Science B.V. All rights reserved
(95)00357-6

mance of one scheme when applied as a single-pass approach only. Solely Valls et al. (1992) provide a direct comparison of the deterministic serial and parallel scheduling scheme. Investigations on sampling applications to solve the RCPSP were reported by Levy et al. (1962) and Wiest (1967). Finally, Alvarez-Valdes and Tamarit (1989b) and Cooper (1976) studied deterministic single-pass *and* probabilistic sampling approaches.

The purpose of this paper is to provide an extensive comparison of the parallel and the serial scheduling scheme. 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 deterministic algorithm of the serial and the parallel scheduling schemes, respectively, as well as their probabilistic extensions. The ability of the scheduling schemes to derive feasible and optimal solutions is covered by observations made in Section 4. Section 5 is devoted to an in-depth computational study based on a set of 360 systematically generated instances. Finally, Section 6 comes up with a summary of the derived results.

2. Problem description

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

$$\text{FT}_i \le \text{FT}_j - d_j \quad j = 2,\dots,J,\ i \in P_j, \tag{2}$$

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

$$\text{FT}_j \ge 0 \quad j = 1,\dots,J. \tag{4}$$

The variable FT_j denotes the (integer valued) finish times of activity j, $j = 1,\dots,J$, and A_t, the set of activities being in progress in period t, is defined as $A_t := \{j \mid j = 1,\dots,J,\ \text{FT}_j - d_j + 1 \le t \le \text{FT}_j\}$. The objective function (1) minimizes 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 (Blazewicz et al. 1983). By relaxing the resource-constraints (3), the RCPSP reduces to the CPM-case (Davis 1966) which can be solved by forward recursion in polynomial time (Elmaghraby, 1977, p. 18 ff.). On account of the inherent intractability of the RCPSP, a multitude of exact and heuristic approaches were proposed.

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

Heuristic approaches for the RCPSP basically involve five different solution methodologies: Single- and multi-pass *priority rule based scheduling* (cf. Section 3), *truncated branch and bound* procedures (Alvarez-Valdes and Tamarit, 1989a), *integer programming based heuristics* (Oguz and Bala, 1994), *disjunctive arc concepts* (Shaffer et al., 1965; Alvarez-Valdes and Tamarit, 1989a; Bell and Han, 1991), and *local search techniques* (Sampson and Weiss, 1993; Leon and Balakrishnan, 1995).

3. Priority rule based scheduling

Although belonging to the oldest solution methodology to solve the RCPSP, priority rule based scheduling is still the most important (heuristic) solution technique. This is due to several reasons: (i) The method is intuitive and easy to use, which makes it highly suitable to be employed within commercial packages. (ii) The method is fast in terms of the computational effort which recommends it to be integrated within local search approaches from artificial intelligence (Storer et al., 1992; Leon and Balakrishnan, 1993). Finally, (iii) multi-pass implementations of the method show the best results obtainable by heuristics today (Kolisch, 1995).

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* method. Both generate 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. In each stage the generation scheme forms the set of all schedulable activities, the so-called decision set. A specific priority rule is then employed in order to choose one or more activities from the decision set which then is scheduled. Note that – within a single pass – each activity is only scheduled once. Both scheduling schemes are presented in detail. For a conceptual comparison with optimal branch and bound based procedures, see Kolisch (1995).

3.1. The serial method

The serial method was proposed by Kelley (1963). It consists of $n = 1,\dots,J$ stages, in each of which one activity is selected and scheduled. Associated with each stage are two disjoint activity-sets: In the *scheduled set* S_n are the activities which were already scheduled and thus belong to the partial schedule. The *decision set* D_n contains the unscheduled activities with every predecessor being in the scheduled set. In each stage one activity from the decision set is selected with a priority rule (in case of ties the activity with the smallest activity number is selected) and scheduled at its earliest precedence and resource feasible start time. Afterwards, the selected activity is removed from the decision set and put into the scheduled set. This, in turn, may place a number of activities into the decision set, since all their predecessors are now scheduled. The algorithm terminates at stage number $n = J$, when all activities are in the partial schedule, i.e. the scheduled set.

To give a formal description of the serial scheduling scheme some additional notation has to be introduced. Let πK_{rt}, the left over capacity of the renewable resource r in period t, and D_n, the decision set, be defined as follows:

$$\pi K_{rt} := K_r - \sum_{j \in A_t} k_{jr},$$

$$D_n := \{ j \mid j \notin S_n,\ P_j \subseteq S_n \}.$$

Further, let EFT_j denote the earliest precedence feasible finish time of activity j within the current partial schedule and let LFT_j denote the latest precedence feasible finish time of activity j as determined by backward recursion from the upper bound of the project's makespan T. Finally, let $\nu(j)$ be a priority value of activity j, $j \in D_n$. Now, the serial scheduling scheme (SSS) can be formally described as follows:

```
[SSS]
**Initialisation**: $n := 1$, $S_n := \emptyset$;
WHILE $|S_n| < J$ DO **Stage** $n$
BEGIN
  COMPUTE $D_n$ and $\pi K_{rt}$, $t = 1,\dots,T$, $r \in R$;
  $j^* := \min_{j \in D_n}\{ j \mid \nu(j) = \inf_{i \in D_n}\{\nu(i)\}\}$;
  $\mathrm{EFT}_{j^*} := \max\{\mathrm{FT}_i \mid i \in P_{j^*}\} + d_{j^*}$;
  $\mathrm{FT}_{j^*} := \min\{t \mid \mathrm{EFT}_{j^*} \le t \le \mathrm{LFT}_{j^*},\ k_{j^*r} \le \pi K_{r\tau},$
      $\tau = t - d_{j^*} + 1,\dots,t,\ r \in R\}$;
  $S_{n+1} := S_n \cup \{j^*\}$;
  $n := n + 1$;
END;
**Stop**
```

Utilising the serial method in a single-pass environment, results were published by Pascoe (1966), Müller-Merbach (1967), Gonguet (1969), Fehler (1969), Cooper (1976, 1977), [1] Boctor (1990), and

[1] Misleadingly, Cooper terms his scheduling scheme to be parallel (serial) when using priority rules in a dynamic (static) fashion. But, as already pointed out by Valls et al. (1992), he clearly employed a serial scheduling scheme.

Valls et al. (1992). Boctor performed a computational study on the basis of 36 small instances from the literature. The two best priority rules employed in his study selected the activity with the smallest final node number in an activity-on-arrow network. Ties were resolved by scheduling the activity with the smallest initial node and with the maximum total resource requirement, respectively. Both rules obtained an average increase of 9.13% above the optimal objective function.

3.2. The parallel method

Today, two algorithms are associated with the so-called *parallel method*: The algorithm of Kelley (1963) and the one of Brooks (Bedworth and 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 and referred to as parallel method.

The parallel method consists of at most J stages in each of which a set of activities (which might be empty) is scheduled. A unique feature of the parallel method is that each stage n is associated with a schedule time t_n, where $t_m \le t_n$ for $m < n$ holds. On account of this schedule time, the set of scheduled activities is now divided into the following two subsets: Activities which were scheduled and are completed up to the schedule time are in the *complete set* C_n, while activities which were scheduled, but which are at the schedule time still active, are in the *active set* A_n. Finally, we have the *decision set* D_n. In contrast to the serial method it contains all yet unscheduled activities which are available for scheduling w.r.t. precedence *and* resource constraints. 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 (again, in case of ties the activity with the smallest label is chosen) and scheduled to start 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 were 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.

Given A_n, the active set, and C_n, the complete set, respectively, πK_r, the left over period capacity of the renewable resource r at the schedule time, and D_n, the decision set, are defined as follows:

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

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

Now, a formal description of the parallel scheduling scheme (PSS) arises to:

```
[PSS]
**Initialisation**: $n := 1,\ t_n := 0,\ D_n := \{1\},\ A_n := C_n := \emptyset,\ \pi K_r := K_r\ \forall r \in R$, GOTO Step (2);
WHILE $|A_n \cup C_n| < J$ DO **Stage** $n$
BEGIN
(1) $t_n := \min\{\mathrm{FT}_j \mid j \in A_{n-1}\}$;
    $A_n := A_{n-1} \setminus \{j \mid j \in A_{n-1},\ \mathrm{FT}_j = t_n\}$;
    $C_n := C_{n-1} \cup \{j \mid j \in A_{n-1},\ \mathrm{FT}_j = t_n\}$;
    COMPUTE $\pi K_r\ \forall r \in R$ and $D_n$;
(2) $j^* := \min_{j \in D_n}\{j \mid \nu(j) = \inf_{i \in D_n}\{\nu(i)\}\}$;
    $\mathrm{FT}_{j^*} := t_n + d_{j^*}$;
    $A_n := A_n \cup \{j^*\}$;
    COMPUTE $\pi K_r\ \forall r \in R$ and $D_n$;
    IF $D_n \neq \emptyset$ THEN GOTO Step (2) ELSE $n := n + 1$;
END;
**Stop**
```

Computational experiments conducted with the single-pass version of the parallel method are more frequent than those with the serial method and are reported by Alvarez-Valdes and Tamarit (1989a,b), Boctor (1990), Davis and Patterson (1975), Elsayed (1982), Lawrence (1985), Pascoe (1966), Patterson (1973, 1976), Thesen (1976), Ulusoy and Özdamar (1989), Valls et al. (1992), and Whitehouse and Brown (1979). Additionally, Arora and Sachdeva (1989) report about an implementation of the parallel method on parallel processors. Davis and Patterson

(1975) document an average increase above the optimum of 5.6% for the minimum slack priority rule and 6.7% for the latest finish time priority rule, when applied to 83 of the instances employed in Patterson (1984). On the basis of their 48 test instances with 27 activities each, Alvarez-Valdes and Tamarit (1989a) come up with an average increase above the optimum of 2.89% for the greatest positional rank priority rule and 3.09% for the latest finish time priority rule.

The only comparison of the serial and the parallel scheduling scheme when applied as deterministic single-pass heuristic is reported by Valls et al. (1992). They concluded that none of the schemes is dominant which contradicts the assumption made by Alvarez-Valdes and Tamarit (1989a) that parallel algorithms "seem to work better than the serial ones".

3.3. Sampling

The way the serial and the parallel scheduling method have been described so far is termed as *single-pass* approach, i.e. one single pass and one priority rule are employed to derive one feasible solution. *Multi-pass* procedures, on the contrary, perform Z single passes in order to generate a sample of at most Z unique feasible solutions, where the best one is chosen. Basically, two different kinds of multi-pass methods can be distinguished: The *multi-priority rule approach* (Lawrence, 1985; Boctor, 1990; Li and Willis, 1992) employs one scheduling scheme and different priority rules while *sampling* (Levy et al., 1962; Wiest 1967; Cooper, 1976; Alvarez-Valdes and Tamarit, 1989b) makes use of one scheduling scheme and one priority rule. Different schedules are obtained by biasing the selection of the priority rule through a random device. The use of a random device can be interpreted as a mapping

$$\psi : j \in D_n \rightarrow [0, 1] \tag{5}$$

which at stage n assigns to each activity in the decision set D_n a probability $\psi(j)$ of being selected (where $\sum_{j\in D_n}\psi(j) = 1$ holds). Three different methods can be distinguished. (i) *Random sampling* assigns each activity in the decision set the same probability. (ii) *Biased random sampling* biases the probabilities dependent on the priority values of the activities to favour those activities which seem to be a more sensible choice (Baker, 1974, p. 72). In the context of the job shop problem, biased random sampling is usually referred to as probabilistic dispatching (Conway et al., 1967, p. 124; Baker, 1974, pp. 202–206). (iii) A special case of biased random sampling is the utilisation of regret measures for determining the selection probabilities. This was introduced by Drexl (1991) and Drexl and Grünewald (1993) and is referred to as *regret based biased random sampling*. Let a priority rule be defined by the mapping $\nu : j \in D_n \rightarrow \mathbb{R}_{\geq 0}$ which assigns to each activity j in the decision set D_n a priority value $\nu(j)$ and an objective O stating whether the activity of the decision set with the minimum (O = min) or maximum (O = max) priority value is selected. Then, the regret ρ_j compares the priority value of activity j with the worst consequence in the decision set as follows:

$$\rho_j := \begin{cases} \max\limits_{i\in D_n} \nu(i) - \nu(j), & \text{if } O = \min, \\ \nu(j) - \min\limits_{i\in D_n} \nu(i), & \text{if } O = \max. \end{cases} \tag{6}$$

Therewith, the parameterised probability mapping arises to

$$\psi(j) := \frac{(\rho_j + 1)^\alpha}{\sum\limits_{i\in D_n} (\rho_i + 1)^\alpha}. \tag{7}$$

Adding the constant "1" to the regret value ρ_j assures that the selection probability for each activity in the decision set is greater than zero and thus every schedule of the population may be generated. By choice of the parameter α, the amount of bias can be controlled. Associated with an arbitrary large α will be no bias and thus deterministic activity selection on the basis of the employed priority rule (with random selection as a tie breaker) while an α of 0 will give way for random activity selection.

Sampling applications of the serial method are documented by Cooper (1976) while sampling efforts on the basis of the parallel method are reported by Wiest (1967) and Alvarez-Valdes and Tamarit (1989b). Employing a sample size of 100, Cooper (1976) compared deterministic scheduling and biased random sampling with nine different priority rules on one benchmark instance. He concluded that sampling produces results which are (at the 99% level of confidence) at least 7% better than the solutions derived by the deterministic approach. Alvarez-Valdes and Tamarit (1989b) compared a single-pass and a sampling approach on a set of 48 instances with 103 activities each. The sampling approach generated for every instance 100 solutions. For the best (second best) priority rule an average increase above an upper bound of 3.23% (3.45%) when used in the single-pass procedure and 2.31% (1.65%) when used in the sampling procedure is reported.

To the best of our knowledge, no comparison of the serial and the parallel scheduling scheme when applied as sampling procedures is reported in the literature.

3.4. Theoretical results

With the following two observations we will now show that both scheduling methods generate feasible schedules which are optimal for the resource-unconstrained case. Furthermore, we prove that the parallel method generates non-delay-schedules while the serial method constructs active schedules.

Observation 1. For any (feasible) instance of the RCPSP a feasible schedule is derived by each of the two scheduling schemes, because both methods take into account precedence and resource constraints.

Observation 2. For instances which are resource-unconstrained, both methods reduce to a simple forward recursion which yields optimal solutions for the polynomial solvable problem (1), (2), and (4).

Theorem. (*a*) *A schedule S generated with the serial scheduling scheme and any priority rule belongs to the set of active schedules while* (*b*) *a schedule S generated with the parallel scheduling scheme and any priority rule belongs to the set of non-delay schedules.*

Proof. In order to show that the Theorem holds, we need to introduce the notion of local and global left shifts for active and non-delay schedules. For a general classification of schedules for the RCPSP we refer to Sprecher et al. (1994).

Following Sprecher et al. (1994) we can define *local* and *global left shifts* along the following line. Starting with a feasible schedule $S = (\mathrm{FT}_1, \ldots, \mathrm{FT}_j, \ldots, \mathrm{FT}_J)$ we ceteris paribus assign activity j the earlier finish time FT_j'', i.e. we are "left shifting" activity j from FT_j to FT_j'' with $\mathrm{FT}_j'' < \mathrm{FT}_j$. We evaluate the resulting schedules $S'' = (\mathrm{FT}_1, \ldots, \mathrm{FT}_j'', \ldots, \mathrm{FT}_J)$ and all intermediate schedules $S' = (\mathrm{FT}_1, \ldots, \mathrm{FT}_j', \ldots, \mathrm{FT}_J)$ with $\mathrm{FT}_j'' < \mathrm{FT}_j' < \mathrm{FT}_j$. If the resulting schedule and all intermediate schedules are feasible we have performed a local left shift of activity j, if the resulting schedule is feasible and at least one intermediate schedule is infeasible we have performed a global left shift of activity j.

Now, Sprecher et al. (1994) define *active* and *non-delay schedules* as follows. An active schedule is defined as a feasible schedule where none of the activities can be locally or globally left shifted. Contrary, a non-delay schedule is defined as a feasible schedule where none of the sub-activities of the corresponding unit-time-duration schedule (a schedule where each activity j is split into d_j sub-activities with duration "1") can be locally or globally left shifted. Note that by definition the set of non-delay-schedules is a non-proper subset of the set of active schedules.

(a) To prove that any schedule generated by the *serial scheduling scheme* belongs to the set of active schedules we proceed as follows. First we show that the schedule is (at least) active and then we verify that it is not a non-delay schedule. Consider activity j has been selected at stage n. Then, its time window of precedence feasible finish times is restricted from the earliest finish time, i.e. the maximum finish time of its immediate predecessors plus its duration, to its latest finish time. Activity j is now scheduled at the earliest contiguous resource feasible interval of length d_j within its precedence feasible time window. Therefore, a left shift of any activity is not possible and the schedule has to be at least active. Furthermore, it has to be verified that the schedule is not a non-delay schedule. This can be achieved by showing that in the corresponding unit-time-duration schedule at least one of the sub-activities can be locally or globally left shifted. Again, consider that activity j with a duration of $d_j > 1$ has been selected at stage n and, additionally, that there are two contiguous resource feasible intervals in the precedence feasible time window of activity j. The "earlier" one with less than d_j and the "latter" one with d_j

units in length. Then, activity j will be scheduled at the earliest contiguous resource feasible interval of length d_j in its precedence feasible time window. Hence, within the corresponding unit-time-duration schedule at least the first sub-activity emanating from activity j can be globally left shifted. Therefore the schedule is not a non-delay schedule and hence has to be an active schedule.

(b) To prove that any schedule generated by the *parallel scheduling scheme* belongs to the set of non-delay schedules, it must be shown that in the corresponding unit-time-duration schedule none of the sub-activities can be locally or globally left shifted. Assume that in the unit-time-duration schedule the first sub-activity emanating from activity j can be globally or locally left shifted to period t_n. Hence, at stage n of the parallel method, activity j has been in the decision set because as a prerequisite to left shift the first sub-activity of j all predecessors of j had to be finished and each resource had to provide enough left over capacity to process activity j. In addition, stage n has been finished without scheduling activity j, leaving enough left over capacity in period t_n to accommodate the first sub-activity of j. Therefore, activity j still has been in the decision set when stage n had been finished. But this is not possible because the algorithm terminates a stage only when the decision set is empty. This assures that the first sub-activity of j cannot be left shifted at all which limits the resulting schedule to be a non-delay schedule. □

At this juncture, it has to be recalled that the set of non-delay schedules might not contain a schedule which optimises a regular measure of performance (Sprecher et al., 1994). In other words: The parallel scheduling scheme searches in a smaller solution space than the serial scheduling scheme, but with the severe drawback that, when considering a regular performance measure, the solution space might not contain the optimal solution.

4. Experimental investigation

4.1. Statistical model

In order to study the performance of the scheduling schemes, the following statistical model (with five factors) was employed (Kurtulus and Davis, 1982; Kurtulus and Narula, 1985):

$$\mathrm{DEV}_{abcmno} = \delta(\mathrm{PR}_a, \mathrm{SS}_b, Z_c, \mathrm{NC}_m, \mathrm{RF}_n, \mathrm{RS}_o) + \varepsilon_{abcmno}, \tag{8}$$

$$\mathrm{CPU}_{abcmno} = \theta(\mathrm{PR}_a, \mathrm{SS}_b, Z_c, \mathrm{NC}_m, \mathrm{RF}_n, \mathrm{RS}_o) + \varepsilon_{abcmno}, \tag{9}$$

where DEV_{abcmno} (CPU_{abcmno}) denotes the average deviation from the optimum solution (the average running time in CPU seconds) when the instances with the m-th, n-th, and o-th level of the *problem parameters* NC, RF, and RS are solved with the a-th, b-th, and c-th level of the *procedure parameters* PR, SS, and Z, respectively. For the errors ε it is assumed that they are mutually independent and that each ε is drawn from the same continuous population.

The *procedure parameters* are characterised as follows: PR denotes the priority rule, SS stands for

Table 1
Good priority rules presented in the literature

Priority rule	Paper	O	$\nu(j)$
Most total successors (MTS)	Alvarez-Valdes and Tamarit	max	$\lvert \bar{S}_j \rvert$
Latest start time (LST)	Alvarez-Valdes and Tamarit	min	$\mathrm{LFT}_j - d_j$
Greatest rank positional weight (GRPW)	Alvarez-Valdes and Tamarit	max	$d_j + \Sigma_{i \in S_j} d_i$
Weighted resource utilisation ratio and precedence (WRUP)	Ulusoy and Özdamar	max	$0.7 \lvert S_j \rvert + 0.3\, \Sigma_{r \in R} k_{jr}/K_r$
Latest finish time (LFT)	Davis and Patterson	min	LFT_j
Minimum slack (MSLK)	Davis and Patterson	min	$\mathrm{LFT}_j - \mathrm{EFT}'_j$

the scheduling scheme while Z denotes the sample size. Priority rules were chosen according to the studies of Davis and Patterson (1975), Alvarez-Valdes and Tamarit (1989a), Valls et al. (1992), Uluzoy and Özdamar (1989), and Boctor (1990). Table 1 provides an overview of the six priority rules which rank among the top three rules in at least one of these studies. Note that only the studies by Boctor (1990) and Valls et al. (1992) employed the (parallel and the) serial scheduling scheme and that none of the three best rules were applied within the serial scheduling scheme. Additional to the notation already introduced, $\bar{S}_j(S_j)$ denotes the set of all (immediate) successors of activity j and EFT'_j denotes the earliest precedence and resource feasible finish time of activity j. Note that in the parallel scheduling scheme EFT'_j equals $t_n + d_j$ for each activity in the decision set. Since within step (2) of the parallel scheduling scheme t_n is constant for all $j \in D_n$, $\nu(j)$ is equal for MSLK and LST which was proven by Davis and Patterson (1975). Further note that LST and MSLK are listed separately because the priority rules are employed in both schemes.

Two levels of the scheduling schemes, i.e. the parallel scheduling scheme (PSS) and the serial scheduling scheme (SSS), were considered. The levels of the sample size are provided in the subsequent sections.

The *problem parameters* are characterised as follows (for details cf. Kolisch et al., 1995): The network complexity NC is the ratio of non-redundant precedence relations to the number of activities. The resource factor RF reflects the density of the two dimensional array k_{jr}, $j = 2, \ldots, J-1$ and $r = 1, \ldots, |R|$. Finally, the resource strength RS measures the degree of resource-constrainedness in the interval [0, 1]. The resource strength is computed as follows: $\mathrm{RS} = (K_r - K_r^{\min})/(K_r^{\max} - K_r^{\min})$, where $K_r^{\min}$ is the minimal availability of resource type r in order to assure feasibility of the RCPSP, i.e. $K_r^{\min} = \max\{k_{jr} \mid j = 1, \ldots, J\}$, and $K_r^{\max}$ is the peak demand of resource type r in a CPM schedule.

All other problem parameters were adjusted as follows (where intervals consist of uniformly distributed integers): The number of non-dummy activities was set to 30, i.e. $J = 32$, the number of resource types was set to 4, i.e. $|R| = 4$, the activity duration was drawn from the interval [1, 10]. In case of a positive resource demand, i.e. $k_{jr} > 0$, the latter was drawn from the interval [1, 10]. The number of different resource types requested by one activity was – depending on the resource factor RF – in the range [1, 4]. Finally, the precedence network was generated with the following constraints: The number of immediate successors (predecessors) of the dummy-source (dummy-sink) was set to 3, respectively, and the number of successors (predecessors) of each non-dummy activity was drawn from the interval [1, 3].

Table 2
Levels of variable problem parameters

NC_m	RF_n	RS_o
{1.5, 1.8, 2.1}	{0.25, 0.5, 0.75, 1}	{0.2, 0.5, 0.7}

In order to generate instances with these problems parameters, ProGen – an instance generator for a broad class of precedence and resource-constrained (project) scheduling problems (Kolisch et al., 1995) – was employed as follows: As shown in Table 2, 10 instances for each combination of NC, RF, and RS were generated which equalled a total of $3 \cdot 4 \cdot 3 \cdot 10 = 360$ problems. Imposing a time limit of 3600 CPU seconds, for 308 of these instances the optimal solution was obtained with the exact procedure of Demeulemeester and Herroelen (1992) on a personal computer with 80386sx processor, mathematical coprocessor and 15 MHz clockpulse. Hence, each of these 308 problems was treated by every level combination of the procedure parameters, scheduling scheme, priority rule, and sample size.

Since it is not confirmed that DEV is normally distributed, only nonparametric tests, namely the Wilcoxon signed rank test and the Friedman test, were employed on a one-way layout without replications (Alvarez-Valdes and Tamarit, 1989a; Golden and Steward, 1985). Confidence levels are denoted with s. Observations with confidence levels of less than or equal to 1%, i.e. $s \leq 0.01$, will be judged as significant. The one-way layout was derived by simply averaging over all factors except the one under consideration (Kurtulus and Davis, 1982).

All algorithms were 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

Table 3
Performance of priority rules

PR_a	LST	LFT	MTS	MSLK	GRPW	WRUP
DEV_a	5.06	5.32	6.65	7.53	10.78	11.66
CPU_a	0.02	0.02	0.03	0.03	0.02	0.03

zu Kiel. Random numbers were drawn with the generator proposed by Schrage (1979).

4.2. Single-pass analysis

For the single-pass analysis the sample size Z_c was set to "1" and the selection of activities was performed deterministically. Table 3 gives a comparison of the priority rules. The Friedman test reveals a significant different performance w.r.t. the average deviation from the optimal solution ($s = 0.0000$). By pairwise application of the Wilcoxon test the following ranking is observed (where "$\succ$" denotes better and "$\succ\succ$" denotes significant better): LST $\succ$ LFT $\succ\succ$ MTS $\succ$ MSLK $\succ\succ$ GRPW $\succ\succ$ WRUP. Thus, four groups can be distinguished significantly ($s \leq 0.0051$): The (lower bound based) rules LST and LFT which perform quite good, MTS and MSLK ranging in the middle, as well as GRPW and finally WRUP, the two latter revealing a poor performance, respectively. The computational effort for all rules is very modest. Within both scheduling schemes, the rules MTS and WRUP require slightly more CPU time, whereas the MSLK rule demands only within the serial scheduling scheme more CPU time. This is because the earliest precedence and resource feasible finish time has to be determined for every activity in the decision set.

Table 4 demonstrates the performance of the scheduling schemes. The ranking reveals to be PSS $\succ\succ$ SSS and hence confirms the conjecture that the parallel scheduling scheme is significantly ($s = 0.0000$) superior to the serial one when used as single-pass heuristic (Alvarez-Valdes and Tamarit, 1989a). Each priority rule performs better w.r.t. the quality of solutions when applied in the parallel scheme. Even more, every priority rule with the exception of the lower bound based rules LST and LFT is significantly better within the parallel scheduling scheme ($s \leq 0.0013$). Nevertheless, since LST and LFT belong to the best rules in both schemes, the serial scheduling scheme cannot be excluded a priori. This conclusion was already drawn in the study by Valls et al. (1992).

Table 4
Performance of scheduling schemes

SS_b	PSS	SSS
DEV_b	6.46	9.21
CPU_b	0.02	0.03

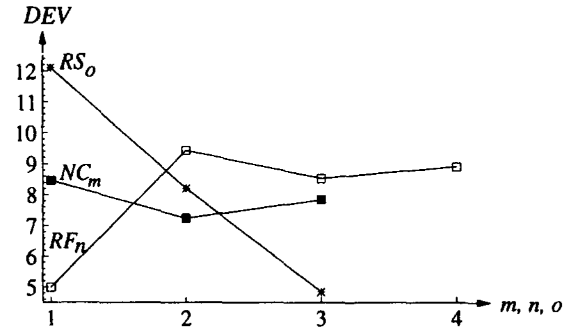

Fig. 1. Effect of the problem parameters on the overall performance.

The running time of the parallel scheduling scheme is slightly less than the one of the serial scheme. Whereas the parallel scheme uses most of the time to update the decision set, the serial scheme requires the majority of the CPU time for setting up and managing the array πK_{rt}, $r \in R$, $t = 1,\dots,T$, which is (especially in the case of a poor upper bound for the makespan T) very time consuming.

The effect of the problem parameters on the overall performance is as follows (cf. Fig. 1 with monotonically increasing parameter levels on the x-axis): The network complexity NC does not reveal a significant influence ($s = 0.2929$), whereas the effect of the resource factor RF and resource strength RS, respectively, is highly significant ($s = 0.0000$). Based on the benchmark-instances of Kolisch et al. (1992), we can conclude that the employed problem parameters influence single-pass scheduling schemes in the same manner as optimal branch-and-bound-based procedures. That is, a high resource factor and a low resource strength will generally induce a poor performance.

A bell-shaped performance w.r.t. the resource parameters as originally proposed in Elmaghraby and

Herroelen (1980) and later experimentally detected in the elaborate study of De Reyck and Herroelen (1993) was not perceived. De Reyck and Herroelen too generated instances with ProGen. But for the analyse of the results they employed two different parameters, namely the complexity index (CI) and the resource constraidness (RC). The CI was introduced by Bein et al. (1992). Essentially, it measures how nearly series-parallel an activity-on-arc network is. The RC was proposed by Patterson (1976) as follows: $RC_r = D_r/K_r$ where D_r denotes the average demand of resource type r, i.e. $D_r = \Sigma k_{jr}/\Sigma\{1,$ if $k_{jr} > 0$; 0 else}. De Reyck and Herroelen found out that keeping CI constant the computation time of the exact procedure of Demeulemeester and Herroelen (1992) reveals a bell-shaped performance for ascending RC values. Hence, it is conjectured that the absence of the bell-shaped curve in this study is caused by not keeping CI constant and/or not systematically varying RC.

Neither the ranking of priority rules nor scheduling schemes is significantly influenced by the problem parameters. Regarding priority rules, this outcome confirms the conclusion made in the studies by Cooper (1976) and Alvarez-Valdes and Tamarit (1989a). Furthermore, the computational effort is not influenced by any of the problem parameters.

4.3. Sampling analysis

In the following it is investigated if the conclusions which were drawn for the single-pass case are still valid in the case of sampling. According to preliminary computational results (Kolisch, 1995), the levels of the sample size Z_c were chosen to be {10, 40, 70, 100} and the bias parameter α was exclusively set to "1".

Table 5 reveals the performance of the priority rules. As for the single-pass case, a significant difference between rules can be detected ($s = 0.0024$). The (lower bound based) rules LST and LFT perform best, MTS and MSLK range in the middle while WRUP and GRPW have the worst performance. The results of MSLK have to be interpreted with care because – like for the single-pass approach – it shows a quite different performance within each of the two scheduling schemes, respectively: For the parallel scheme it performs like the "good" LST rule, within the serial scheme MSLK gives rather poor results. For sampling the computational requirement of all priority rules increases linearly to an average of 1.4 CPU seconds. This is very moderate compared to the average time of 45.24 CPU seconds needed by the Demeulemeester and Herroelen algorithm in order to solve all problems to optimality. But using the latter procedure as truncated branch-and-bound method with a maximum of 1 CPU second, it achieves a deviation of 1.21% and thus outperforms each of the priority rules. Hence, when employing sampling in a competitive and efficient manner, one has to make use of bounding rules in order to cut down the computational effort (Cooper, 1974; Kolisch, 1995).

By comparing the ranking (LFT $\succ$ LST $\succ\succ$ MTS $\succ$ MSLK $\succ\succ$ WRUP $\succ\succ$ GRPW) with the one of the single-pass approach the following can be stated: (i) With two exceptions (LST and LFT as well as WRUP and GRPW) the ranking obtained is the same as for the single-pass case. (ii) While the difference between groups (i.e. LFT, LST vs. MTS, MSLK vs. WRUP vs. GRPW) has about the same level of significance ($s \leq 0.0036$), the difference between rules is slightly less significant. Thus, it can be stated that, in general, *priority rules which are good for single-pass approaches are good for biased random sampling approaches and vice versa.* Although not explicitly pointed out, Alvarez-Valdes and Tamarit (1989b) attained similar results in their study. Their rankings obtained when utilising six priority rules within the deterministic- and the sampling-based parallel scheduling scheme differed only w.r.t. one rule. The opposite observation by Cooper (1976) is a consequence of his (not regret based) probability mapping which in conjunction with the LFT priority rule tends to perform (pure) random sampling. Of course, the interrelation between single-pass and sampling heuristics depends on the amount of bias: Whereas with 0% bias the significance equals the one of the (deterministic) single-pass case, a bias of

Table 5
Performance of traditional priority rules for sampling

PR_a	LFT	LST	MTS	MSLK	WRUP	GRPW
DEV_a	2.08	2.11	2.48	2.56	3.36	3.79
CPU_a	1.12	1.11	1.68	1.63	1.17	1.65

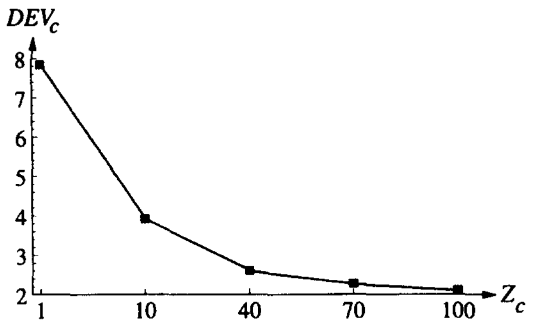

Fig. 2. Performance as a function of the sample size.

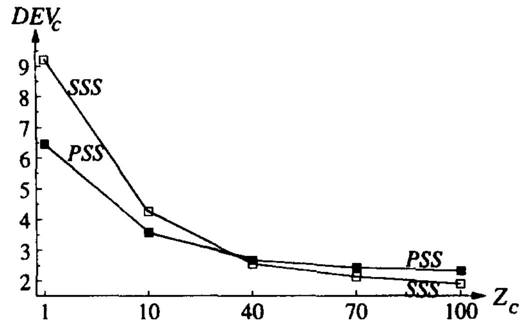

Fig. 3. Impact of the sample size on the performance of the scheduling schemes.

100% (i.e. random sampling) results in no (significant) difference between the priority rules.

Fig. 2 gives insight into the performance w.r.t. the sample size. As expected, it is demonstrated that increasing the sample size continuously produces better solutions. Depending on the sample size, the average performance of the single-pass approach is thus improved between 50% ($Z_c = 10$) and 73% ($Z_c = 100$). This is up to ten times more than observed by Cooper (1976). Hence, it can be stated that *sampling significantly outperforms the single-pass approach* ($s = 0.0000$). This contradicts the conclusions drawn by Conway et al. (1967, p. 128) for the job shop problem, stating that sampling reveals only modest improvement over single-pass procedures. But it has to be noted that the marginal improvement diminishes. This implies a growing computational effort in order to produce better solutions.

Finally, Fig. 3 demonstrates the effect of the scheduling schemes w.r.t. the sample size Z_c. The overall performance of both schemes is with 2.75% for the parallel and 2.71% for the serial scheduling scheme almost identical. But a second glance reveals a (not significant ($s \geq 0.1994$)) different performance w.r.t. the sample size. While *the parallel scheme is clearly superior for small sample sizes* (i.e. less than 40 generated schedules), *the serial scheme shows better results for large samples.* Consequently, for sampling procedures solving the RCPSP the general superiority of non-delay schedules, as announced in Conway et al. (1967, pp. 121–124) for the job shop problem when minimising the average flow time, does not hold true. The rationale of the observation is as follows: For small sample sizes the superiority of the parallel scheme w.r.t. the single-pass approach is dominant. With increasing sample size, this effect diminishes and at the same time the parallel scheme suffers from the fact that the sample space is the set of non-delay schedules which not necessarily contains the optimal solution. We conjecture that this "critical sample size", i.e. 40 for the instances tested, increases when the problem size, expressed in the number of activities, is enlarged.

In order to investigate how much the parallel scheduling scheme is restricted by operating on the set of non-delay schedules, a full enumeration of all schedules in this set was performed as follows: In step (2) of PSS a branch was created for each non-dominated set of activities out of D_n which could be jointly started at t_n. Imposing a time limit of 24 CPU hours, the best non-delay schedule was obtained for 298 of the 308 instances. Table 6 shows the frequency distribution of the deviation for the best non-delay schedules. For 123 problems, i.e. 41.27%, the set of non-delay schedules did not contain the optimal solution.

The effect of the three problem parameters network complexity, resource factor, and resource

Table 6
Frequency distribution of the deviation for the best non-delay schedules

DEV range	0	(0, 1]	(1, 2]	(2, 3]	(3, 4]	(4, 5]	(5, 10]	> 10
Number of instances	175	1	33	25	19	14	26	5

Table 7
Effect of RF and RS on the ranking of scheduling schemes

	$RS_1 = 0.2$	$RS_2 = 0.5$	$RS_3 = 0.7$	
$RF_1 = 0.25$	SS	SS	SS *	SS *
$RF_2 = 0.5$	PS	SS	SS *	SS
$RF_3 = 0.75$	PS	PS *	PS	PS *
$RF_4 = 1$	PS *	PS *	PS	PS *
	PS	PS *	SS *	

strength on the average performance turned out to be as for the single-pass case. That is, no significant influence can be observed for the network complexity ($s = 0.6485$) while resource strength and resource factor, in the order mentioned, turned out to be highly significant ($s = 0.0000$). The ranking of the priority rules is not significantly effected by the problem parameters but, deviating from the single-pass case, resource factor and resource strength influence the ranking of scheduling schemes significantly.

Table 7 shows the ranking of the scheduling schemes w.r.t. both resource parameters as well as their combination (where * denotes a 1-tailed significance at the 1% level of confidence and the bold entries signal superiority of the serial method). Roughly, it can be stated that *the parallel scheduling scheme performs better for "hard" problems* (with a high resource factor and/or a low resource strength) while *the serial scheduling scheme is better for "easy" problems* (with a low resource factor and/or a high resource strength).

5. Summary

We reviewed two scheduling schemes, the serial and the parallel method, by giving a formal description and an extensive literature review. Both methods generate feasible schedules which are optimal in the absence of resource restrictions. It was proven that the serial method generates active schedules while the parallel scheduling scheme creates non-delay schedules. Hence, the parallel scheduling scheme searches in a smaller solution space than the serial scheduling scheme but with the drawback that – when considering a regular performance measure – the solution space might not contain an optimal schedule.

An in-depth computational study with the instance-set of Kolisch et al. (1995) produced the following results when using both methods as a frame for single-pass scheduling and sampling: Priority rules which derive good results for single-pass scheduling do so for sampling. When performed properly, i.e. when pure random sampling is avoided, sampling improves the results of single-pass scheduling up to 70%. Deviating from the general belief, we showed that the parallel method does not generally perform better than the serial method. Rather, it gives only better results for single-pass scheduling and small sample sizes as well as for "hard" (that is highly resource-constrained) problems. Hence, the serial method is superior for large sample sizes and for instances which are only moderately resource-constrained. This insight should be of importance when deriving fast problem-specific parameter-guided heuristics.

Acknowledgement

The author is indebted to Erik Demeulemeester, Katholieke Universiteit Leuven, for providing the code of his algorithm, to Andreas Drexl, Universität Kiel, for his help in this research, and to two anonymous referees for their helpful comments and suggestions on an earlier version of this paper.

References

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

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

Arora, R.K., and Sachdeva, (1989), "Distributed simulation of resource constrained project scheduling", *Computers & Operations Research* 16, 295–304.

Baker, K.R. (1974), *Introduction to Sequencing and Scheduling*, Wiley, New York.

Balas, E. (1971), "Project scheduling with resource constraints", in: E.M.L. Beale (Ed.), *Applications of Mathematical Programming techniques*, English University Press, London, 187–200.

Bedworth, D.D., and Bailey, J.E. (1982), *Integrated Production Control Systems – Management, Analysis, Design*, Wiley, New York.

Bein, W.W., Kamburowski, J. and Stallmann, M.F.M. (1992), "Optimal reduction of two-terminal directed acyclic graphs", *SIAM Journal on Computing* 21, 1112–1129.

Bell, C.E., and Han, S. (1991), "A new heuristic solution method in resource-constrained project scheduling", *Naval Research Logistics* 38, 315–331.

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

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

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

Bowman, E.H. (1959), "The schedule-sequencing problem", *Operations Research* 7, 621–624.

Carlier, J., and Latapie, B. (1991), "Une methode arborescente pour resoudre les problemes cumulatifs", *Recherche operationnelle* 25, 311–340.

Carruthers, J.A., and Battersby, A. (1966), "Advances in critical path methods", *Operational Research Quarterly* 17, 359–380.

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

Conway, R.W., Maxwell, W.L., and Miller, L.W. (1967), *Theory of Scheduling*, Addison-Wesley, Reading, MA.

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

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

Davis, E.W. (1966), "Resource allocation in project network models – A survey", *The Journal of Industrial Engineering* 17, 177–188.

Davis, E.W., and Heidorn, G.E. (1971), "An algorithm for optimal project scheduling under multiple resource constraints", *Management Science* 17, 803–816.

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

De Reyck, B., and Herroelen W.S. (1996), "On the use of the complexity index as a measure of complexity in activity networks II", *European Journal of Operational Research*, to appear.

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

Drexl, A. (1991), "Scheduling of project networks by job assignment", *Management Science* 37, 1590–1602.

Drexl, A., and Grünewald J. (1993), "Nonpreemptive multi-mode resource-constrained project scheduling", *IIE Transactions* 25/5, 74–81.

Elmaghraby, S.E. (1977), *Activity Networks: Project Planning and Control by Network Models*, Wiley, New York.

Elmaghraby, S.E., and Herroelen, W.S. (1980), "On the measurement of complexity in activity networks", *European Journal of Operational Research* 5, 223–234.

Elsayed, E.A. (1982), "Algorithms for project scheduling with resource constraints", *International Journal of Production Research* 20, 95–103.

Fehler, D.W. (1969), "Die Variationen-Enumeration – Ein Näherungsverfahren zur Planung des optimalen Betriebsmitteleinsatzes bei der Terminierung von Projekten", *Elektronische Datenverarbeitung* 10, 479–483.

Golden, B.L., and Steward, W.R. (1985), "Empirical analysis of heuristics", in: E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys (Eds.), *The Traveling Salesman Problem*, Wiley, New York, 207–249.

Gonguet, L. (1969), "Comparison of three heuristic procedures for allocating resources and producing schedules", in: H.J.M. Lombaers (Ed.), *Project Planning by Network Analysis*, North-Holland, Amsterdam, 249–255.

Hastings, N.A.J. (1972), "On resource allocation in project networks", *Operational Research Quarterly* 23, 217–221.

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

Kolisch, R. (1995), *Project Scheduling under Resource Constraints – Efficient Heuristics for Several Problem Classes*, Physica, Heidelberg.

Kolisch, R., Sprecher, A., and Drexl, A. (1995), "Characterization and generation of a general class of resource-constrained project scheduling problems", *Management Science* 41, 1693–1703.

Kurtulus, I.S., and Davis E.W. (1982), "Multi-project scheduling: Categorization of heuristic rules performance", *Management Science* 28, 161–172.

Kurtulus, I.S., and Narula, S.C. (1985), "Multi-project scheduling: Analysis of project performance", *IIE Transactions* 17, 58–66.

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, PA.

Leon, V.J., and Balakrishnan, R. (1995), "Strength and adaptability of problem-space based neighborhoods for resource constrained scheduling", *OR Spektrum* 17, 173–182.

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

Li, R.K.-Y., and Willis, J. (1992), "An iterative scheduling technique for resource-constrained project scheduling", *European Journal of Operational Research* 56, 370–379.

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

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

Pascoe, T.L. (1966), "Allocation of resources C.P.M.", *Revue Francaise Recherche Operationelle* 38, 31–38.

Patterson, J.H. (1973), "Alternate methods of project scheduling with limited resources", *Naval Research Logistics Quarterly* 20, 767–784.

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

Patterson, J.H. (1984), "A comparison of exact approaches for solving the multiple constrained resource, project scheduling problem", *Management Science* 30, 854–867.

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

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

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

Radermacher, F.J. (1985/86), "Scheduling of project networks", *Annals of Operations Research* 4, 227–252.

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

Schrage, L. (1979), "A more portable Fortran random number generator", *ACM Transactions on Mathematical Software* 5, 132–138.

Shaffer, L.R., Ritter, J.B., and Meyer, W.L. (1965), *The Critical-Path Method*, McGraw-Hill, New York.

Sprecher, A., Kolisch, R., and Drexl, A. (1995), "Semi-active, active, and non-delay schedules for the resource-constrained project scheduling problem", *European Journal of Operational Research* 80, 94–102.

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

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

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

Thesen, A. (1976), "Heuristic scheduling of activities under resource and precedence restrictions", *Management Science* 23, 412–422.

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

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

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

Wiest, J.D. (1967), "A heuristic model for scheduling large projects with limited resources", *Management Science* 13, B359–B377.