Schirmer, Andreas; Riesenberg, Sven

Parameterized heuristics for project scheduling: Biased random sampling methods

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 456

Manuskripte
aus den
Instituten für Betriebswirtschaftslehre
der Universität Kiel

No. 456

Parameterized Heuristics for Project Scheduling - Biased Random Sampling Methods

Schirmer, Riesenberg

September 1997

Andreas Schirmer, Sven Riesenberg
Institut für Betriebswirtschaftslehre, Lehrstuhl für Produktion und Logistik
Christian-Albrechts-Universität zu Kiel, Olshausenstr. 40, D-24118 Kiel, Germany

Phone, Fax +49-431-880-15 31
E-Mail schirmer@bwl.uni-kiel.de

http://www.wiso.uni-kiel.de/bwlinstitute/prod
ftp://www.wiso.uni-kiel.de/pub/operations-research

Contents

Figures

Tables

Abstract: The resource-constrained project scheduling problem is notoriously intractable. Due to its complexity, the majority of algorithms are heuristics, of which priority rule-based methods constitute the most important class. Of these, parameterized biased random sampling methods outperform all other priority rule-based methods known. Different random sampling schemes have been proposed; of these the regret-based scheme (RBRS) of Drexl (1991) is currently the best-performing one. Yet, careful analysis reveals each of these schemes to possess some immanent flaw whose effects may lead to significant distortions of the scheduling process. Thus, we propose some new sampling schemes, completely avoiding some of the effects while grossly reducing others. We discuss a comprehensive experimental evaluation of both classical and new schemes; in addition, we substantially extend the analysis on serial and parallel scheduling heuristics of Kolisch (1996b). Our results indicate that some of the new schemes offer significant improvements over existing sampling algorithms, the best-performing one improving upon the RBRS by more than ten percent. Also, we expose detailed insight into which parameter settings are the most beneficial for different algorithmic schemes.

Keywords: HEURISTICS; PRIORITY RULES; PROJECT SCHEDULING

1. Introduction

The resource-constrained project scheduling problem (RCPSP) is notoriously intractable. Its optimization variant is known to be strongly **NP**-hard, even strongly **NP**-equivalent (Schirmer 1996); indeed, assuming a deadline to be given for the project completion even its feasibility variant is strongly **NP**-complete (Garey, Johnson 1975). Consequentially, if one insists on algorithms guaranteed to find an optimal solution, the current state of the art still has only exponential answers to offer. These include implicit enumeration methods (Talbot, Patterson 1978; Christofides et al. 1987; Alvarez-Valdés, Tamarit 1989; Tavares 1990; Demeulemeester, Herroelen 1992, 1995; Mingozzi et al. 1994; Sprecher 1996), zero-one programming (Bowman 1959; Pritsker et al. 1969; Patterson, Huber 1974; Patterson, Roth 1976), and dynamic programming (Carruthers, Battersby 1966). Yet, considering the complexity of the RCPSP, it comes as no surprise that the majority of algorithms devised are heuristic in nature. From their wealth, construction methods (priority rule-based algorithms, disjunctive arc methods), iterative improvement methods (local search, genetic algorithms), and incomplete exact methods (truncated branch-and-bound) have been developed. No approximation algorithms are reported in the literature that we are aware of. Detailed reviews of algorithms for the RCPSP are given e.g. in Herroelen (1972); Davis (1973); Patterson (1984); Kolisch, Padman (1997).

Of the heuristics proposed, priority rule-based methods despite their age still constitute the most important class of scheduling methods. Kolisch (1996b) names several reasons for their popularity. First, they are straightforward and easy to use which makes them easy to implement. Actually, most commercial scheduling software relies on simple priority rules (De Wit, Herroelen 1990; Kolisch 1997). Second, they are inexpensive in terms of computer time and memory required which makes them amenable even for large instances of computationally intractable problems. Third, their inexpensiveness allows to integrate them as fast "subroutines" into more complex metaheuristic algorithms (Storer et al. 1992; Bean 1994; Leon, Balakrishnan 1995; Naphade et al. 1995; Lee, Kim 1996; Özdamar 1996; Hartmann 1997).

Of these in turn, parameterized biased random sampling methods have recently attracted particular interest, due to the fact that they currently outperform all other priority rule-based methods known, in particular deterministic ones (Kolisch 1996b). Different random sampling schemes have been proposed of which the regret-based scheme (RBRS) of Drexl (1991) is the best-performing one currently known. Careful analysis, however, reveals each of these schemes to possess some immanent flaw whose effects may lead to significant distortions of the scheduling process. We therefore propose some new sampling schemes, each of which completely avoids some of the adverse effects while grossly reducing others. The results from a comprehensive experimental evaluation of both classical and new schemes are discussed; in addition, we substantially extend the analysis on serial and parallel scheduling heuristics presented in Kolisch (1996b). Our results indicate that some of the new schemes offer significant improvements over existing sampling algorithms, the best-performing one improving upon the RBRS by more than ten percent. Also, we expose detailed insight into which parameter settings are the most beneficial for different algorithmic schemes.

This paper is organized along the following lines. The RCPSP is briefly introduced in Section 2. The fundamentals of priority rule-based scheduling, i.e. scheduling schemes and priority rules are covered in Section 3. Random sampling schemes are discussed in Section 4. Section 5 is devoted to the experimentation carried out. Some summarizing remarks in Section 6 conclude the paper.

2. Resource-Constrained Project Scheduling

2.1. Problem Setting

The classical single-mode variant of the resource-constrained project scheduling problem can be characterized as follows: The single project consists of a number of activities of known duration; all activities have to be executed in order to complete the project. During their nonpreemptable execution the activities request renewable resources whose available amount is limited in each period by a constant capacity. Precedence relations between activities stipulate that some activities must be finished before others may be started. Table 1 summarizes the problem parameters of the RCPSP, where w.l.o.g. J, R, d_j, K_r (k_{jr}) are assumed to be positive (nonnegative) integers.

The goal is to find an assignment of periods to activities (a *schedule*) that covers all activities, ensures for each renewable resource r that in each period the total usage of r by all activities performed in that period does not exceed the per-period availability of r, respects the partial order $\angle$, and minimizes the total project length.

Problem Parameter	Definition
d_j	Duration of activity j
J	Number of activities, indexed by j
k_{jr}	Per-period usage of renewable resource r required to perform activity j
K_r	Per-period availability of renewable resource r
R	Number of renewable resources, indexed by r
$\angle$	Partial order on the activities, representing precedence relations

Table 1: Problem Parameters of the RCPSP

2.2. *Model Formulation*

To simplify the formulation of the model, w.l.o.g. it is assumed that the activities 1 and J are *dummy activities*, having durations and resource requirements of zero, and that activity 1 (J) is the unique first (last) activity w.r.t. $\angle$. Also, several parameters are derived from the above problem parameters. First, in order to restrict the number of periods to be considered, let denote T an upper bound for the makespan of the project. Second, let denote P_j ($2 \le j \le J$) the set of all *immediate* predecessors of activity j w.r.t. $\angle$. Third, for each activity j ($1 \le j \le J$) earliest finish times EFT_j and latest finish times LFT_j are calculated (Kelley 1961, 1963). Finally, let denote A_t ($1 \le t \le T$) the set of all (non-dummy) activities being active in period t and T the set of all periods in which at least one activity begins. Then, using integer variables y_j ($1 \le j \le J$) to denote the period in which activity j is finished, the RCPSP can be couched as:

Minimize

$$Z(y) = y_J \tag{1}$$

subject to

$$y_i \le y_j - d_j \qquad (2 \le j \le J;\ i \in P_j) \tag{2}$$

$$\sum_{j \in A_t} k_{jr} \le K_r \qquad (1 \le r \le R;\ t \in T) \tag{3}$$

$$EFT_j \le y_j \le LFT_j \qquad (1 \le j \le J) \tag{4}$$

Minimization of the objective function (1) enforces the earliest possible completion of the last activity J and thus leads to the minimal schedule length. The precedence constraints (2) guarantee that the precedence order is respected while the capacity constraints (3) limit the total resource consumption of each renewable resource in each period to the available amount.

3. Priority-Rule Based Scheduling

Priority rule-based methods consist of at least two components, viz. one (or more) scheduling schemes and one (or more) priority rules. A *scheduling scheme* determines how a schedule is constructed, building feasible *full schedules* (which cover all activities) by augmenting *partial schedules* (which cover only a proper subset of the activities) in a stage-wise manner. On each stage, the scheme determines the set of all activities which are currently eligible for scheduling. In this, *priority rules* serve to resolve conflicts where more than one activity could be feasibly scheduled.

3.1. Scheduling Schemes

Two variants of scheduling schemes are usually distinguished, viz. serial and parallel ones. *Serial scheduling* methods (Kelley 1963) start by numbering the candidates in such a way that no one gets a lower number than any of its predecessors. A schedule is then built by considering the candidates in order and scheduling them one at a time for earliest possible execution w.r.t. the constraints. In contrast, *parallel scheduling* methods[1] proceed by considering the periods of the planning horizon in chronological order. In each period t, of those activities that may feasibly commence in t as many as possible are scheduled one at a time for starting in t.

3.1.1. Serial Scheduling

The *serial scheduling scheme* (SSS) divides the set of activities into three disjoint subsets or states: *scheduled, eligible,* and *ineligible* (cp. Kurtulus, Narula 1985). An activity that is already in the partial schedule is *scheduled.* Otherwise, an activity is called *eligible* if all its predecessors are scheduled, and *ineligible* otherwise. The scheme proceeds in N = J stages, indexed by n. For notational purposes, we refer on stage n to the set of scheduled activities as S_n and to the set of eligible activities as *decision set* D_n. D_n is determined dynamically from

$$D_n \leftarrow \{j \mid j \notin S_n \wedge P_j \subseteq S_n\} \qquad (1 \le n \le N) \quad (5)$$

On each stage n, one activity j from D_n is selected - using a priority rule if more than one activity is eligible - and scheduled to begin at its earliest feasible start time. Then j is moved from D_n to S_n which may render some ineligible activities eligible if now all their predecessors are scheduled. The scheme terminates on stage N when all activities are scheduled.

For a formal description of the SSS let denote RK_{rtn} ($1 \le r \le R$; $1 \le t \le T$; $1 \le n \le N$) the remaining capacity of resource r in period t on stage n and LST_j the latest start time of activity j

---

[1] Usually ascribed to Kelley (1963), they were actually introduced in their common form in an unpublished paper by Brooks in (1963), cf. the remarks in Bedworth, Bailey (1982); Kolisch (1995, p. 68).

($1 \leq j \leq J$). Then the SSS can be formulated as in Table 2 (the details of selecting a job according to (7) are discussed below), where $EPST_j$ ($EFST_j$) denotes the earliest precedence-feasible (feasible, i.e. precedence- and resource-feasible) start time of activity j.

Initialization

$n \leftarrow 1$
$FT_1 \leftarrow 0$
for each $(1 \leq r \leq R; 1 \leq t \leq t)$
    $RK_{rt1} \leftarrow K_r$
$S_n \leftarrow \{1\}$

Execution

for[2] $n \leftarrow 2$ **to** J
    calculate D_n **according to** (5)
    if $D_n = \{j\}$
        $j^* \leftarrow j$
    else
        select j^* **according to** (7)
    $EPST_{j^*} \leftarrow \max \{FT_j \mid j \angle j^*\}$
    $EFST_{j^*} \leftarrow \min \{\tau \mid EPST_{j^*} \leq \tau \leq LST_{j^*} \wedge k_{j^*r} \leq RK_{rtn} \; (1 \leq r \leq R; \tau+1 \leq t \leq \tau+d_{j^*})\}$
    $FT_{j^*} \leftarrow EFST_{j^*} + d_{j^*}$
    $S_n \leftarrow S_n \cup \{j^*\}$
    for each $(1 \leq r \leq R; EFST_{j^*} + 1 \leq t \leq FT_{j^*})$
        $RK_{rtn} \leftarrow RK_{rtn-1} - k_{j^*r}$
endfor

Result

For each activity $j \in S_n$, FT_j denotes the period in which j is finished. ■

Table 2: Serial Scheduling Scheme

The SSS has been studied, among others, by Pascoe (1966), Müller-Merbach (1967), Cooper (1976)[3], Boctor (1990), Valls et al. (1992), and Kolisch (1996b). Note that the SSS searches among the set of active schedules which always contains all optimal schedules for the RCPSP-instance considered (Sprecher et al. 1995; Kolisch 1996b).

[2] We use the construct "**for each** ..." whenever the order in which the respective variable is instantiated is irrelevant for the algorithmic outcome, and the construct "**for** ... **to** ..." otherwise.

[3] Although Cooper claims his scheduling scheme to be parallel, it is actually a serial one. This seems to have been noted first by Valls et al. (1992).

3.1.2. Parallel Scheduling

The *parallel scheduling scheme* (PSS) proceeds in $N \leq J$ stages, indexed by n. Each stage n is associated with a schedule time t_n. The scheme divides the set of activities into four disjoint subsets or states: *active*, *finished*, *eligible*, and *ineligible* (cp. Kurtulus, Narula 1985). A scheduled activity is *active* during its execution, afterwards it becomes *finished*. In contrast to the serial scheme, an activity that is neither active nor finished is called *eligible* if it could be scheduled w.r.t precedence *and* resource constraints, *ineligible* otherwise. Consequentially, the partial schedule consists of all active and finished activities. We refer to the set of active (finished, eligible) activities on stage n, i.e. in period $t_n + 1$, as A_n (F_n, D_n). Another difference to the serial scheme is that each stage consists of two steps. First, the schedule time t_n is iteratively set to the minimum of the finish times of all activities in A_{n-1} (Johnson 1967); then activities with a finish time equal to t_n are moved from A_n to F_n which in turn may make some formerly ineligible activities eligible, transferring them to D_n. This process is repeated until the decision set is indeed nonempty. Second, one eligible activity is selected, using a priority rule if more than one activity is eligible, scheduled to begin at the current schedule time, and moved from D_n to A_n; this second step is repeated as long as D_n is nonempty. The scheme terminates when each activity is scheduled, i.e. is either active or finished.

For a formal description of the PSS, let denote RK_{rn} ($1 \leq r \leq R$; $1 \leq n \leq N$) the remaining capacity of resource r at the schedule time t_n. D_n is derived dynamically from

$$D_n \leftarrow \{j \mid j \notin A_n \cup F_n \wedge P_j \subseteq F_n \wedge k_{jr} \leq RK_{rn}\ (1 \leq r \leq R)\} \qquad (1 \leq n \leq N) \quad (6)$$

Now, the PSS can be formulated as in Table 3 (selecting a job according to (7) is discussed below).

Initialization

$n \leftarrow 1$
$FT_1 \leftarrow 0$
$S_n \leftarrow \{1\}$
$t_n \leftarrow 0$
$A_n \leftarrow \varnothing$
$F_n \leftarrow \{1\}$
$D_n \leftarrow \{j \mid 2 \leq j \leq J\text{-}1 \wedge P_j = \{1\}\}$

Execution

call ScheduleActivities
while $|A_n \cup F_n| < J$
  repeat
    $n \leftarrow n+1$
    $t_n \leftarrow \min \{FT_j \mid j \in A_{n-1}\}$
    $A_n \leftarrow A_{n-1} \setminus \{j \in A_{n-1} \mid FT_j = t_n\}$
    $F_n \leftarrow F_{n-1} \cup \{j \in A_{n-1} \mid FT_j = t_n\}$
    for each $(1 \leq r \leq R)$
      $RK_{rn} \leftarrow K_r - \Sigma_{j \in A_n} k_{jr}$
    calculate D_n **according to** (6)
  until $D_n \neq \varnothing$
  call ScheduleActivities
endwhile

Subroutine ScheduleActivities

repeat
  if $D_n = \{j\}$
    $j^* \leftarrow j$
  else
    select j^* **according to** (7)
  $FT_{j^*} \leftarrow t_n + d_{j^*}$
  $S_n \leftarrow S_n \cup \{j^*\}$
  $A_n \leftarrow A_n \cup \{j^*\}$
  for each $(1 \leq r \leq R)$
    $RK_{rn} \leftarrow RK_{rn} - k_{j^*r}$
  calculate D_n **according to** (6)
until $D_n = \varnothing$

Result

For each activity $j \in S_n$, FT_j denotes the period in which j is finished. ■

Table 3: Parallel Scheduling Scheme

The PSS has been examined by numerous authors, among these Pascoe (1966), Davis, Patterson (1975), Patterson (1973, 1976), Whitehouse, Brown (1979), Alvarez-Valdés, Tamarit (1989), Boctor (1990), Ulusoy, Özdamar (1989), Valls et al. (1992), and Kolisch (1996b). Note that the PSS searches among the non-delay schedules which do not necessarily contain any optimal solution of the RCPSP-instance considered (Kolisch 1996b).

3.2. *Priority Rules*

Priority rules allow to resolve conflicts between activities competing for the allocation of scarce resources. In situations where the decision set contains more than one candidate, priority values are calculated from numerical measures which are related to properties of the activities, the complete project, or the incumbent partial schedule (Cooper 1976). Formally, any priority rule can be specified in terms of two components. One is a mapping v: $D_n \rightarrow R_{\geq 0}$ which accords a numerical measure v(j) to each candidate in the decision set. The other is a dichotomical parameter extr $\in$ {max, min} which specifies which extremum of the priority values determines the candidate to be selected. Ties can be broken arbitrarily, e.g. by smallest activity index (as done in the sequel) or randomly. Then, any such (deterministic) selection of an activity j* (as used above in Tables 2 and 3) can be expressed as

$$j^* \leftarrow \min \{ j \in D_n \mid v(j) = \text{extr} \{ v(j') \mid j' \in D_n \} \} \tag{7}$$

3.2.1. *Network- and Time-Based Rules*

We employ a number of priority rules. For a start, let us briefly introduce several well-known network- or time-based ones, related to the *shortest processing time* (SPT), *most total successors* (MTS), *latest start* or *finish time* (LST, LFT), (static) *slack* (SLK), *greatest rank positional weight* (GRPW), and *weighted resource utilization ratio and precedence* (WRUP)[4]. Also, we complement the *resource scheduling method* (RSM) by two recent modifications, namely the *improved resource scheduling method* (IRSM) and *worst-case slack* (WCS). All these rules were selected since they have been found to be among the best-performing ones for the problem at hand in several studies (Davis, Patterson 1975; Alvarez-Valdés, Tamarit 1989; Ulusoy, Özdamar 1989; Boctor 1990; Kolisch 1996a).

Let for each activity j ($1 \leq j \leq J$) denote S_j the set of all *immediate* successors w.r.t. $\angle$ and $EFST_j$ ($1 \leq j \leq J$) the - dynamically updated - earliest feasible start time w.r.t. *all* constraints[5]. For the measures $IRSM_j$ and WCS_j, let denote AP_n the set of all pairs of nonidentical activities i and j in the decision set and E_{ij} the earliest time to schedule activity j if activity i is started at t_n (for details cf. Kolisch 1996a). Using this notation, the rules can be defined as done in Table 4 where also a classification in terms of several straightforward criteria is given; the last criterion refers to whether the rule is applicable in both scheduling schemes or only in the parallel one (Cooper 1976; Kolisch 1995, pp. 85-86).

---

[4] GRPW is described in Alvarez-Valdés, Tamarit (1989). WRUP is introduced by Ulusoy, Özdamar (1989); in our implementation, we used the setting $w_1 = 0.7$ and $w_2 = 0.3$ which the authors report as producing the best results. RSM is due to Brand et al. (1964), IRSM as well as WCS to Kolisch (1996a).

[5] In the PSS, using $EFST_j$ as a measure would be equivalent to SPT_j (Valls et al. 1992).

Extremum	Measure	Definition	Static vs. Dynamic	Local vs. Global	Simple vs. Composite	Serial vs. Parallel
MIN	SPT_j	$\leftarrow d_j$	S	L	S	S, P
MAX	MTS_j	$\leftarrow \lvert\{j' \mid j \angle j'\}\rvert$	S	L	S	S, P
MIN	LST_j	$\leftarrow LFT_j - d_j$	S	G	S	S, P
MIN	LFT_j	$\leftarrow LFT_j$	S	L	S	S, P
MIN	SLK_j	$\leftarrow LST_j - EFST_j$	D	L	S	S, P
MAX	$GRPW_j$	$\leftarrow d_j + \Sigma_{i \in S_j} d_i$	S	L	S	S, P
MAX	$WRUP_j$	$\leftarrow w_1 \lvert S_j \rvert + w_2 \Sigma_r k_{jr}/K_r$	S	L	C	S, P
MIN	RSM_j	$\leftarrow \max\{0, t_n + d_i - LST_i \mid (i,j) \in AP_n\}$	D	G	C	P
MIN	$IRSM_j$	$\leftarrow \max\{0, E_{ji} - LST_j \mid (i,j) \in AP_n\}$	D	G	C	P
MIN	WCS_j	$\leftarrow LST_j - \max\{E_{ij} \mid (i,j) \in AP_n\}$	D	G	C	P

Table 4: Network- and Time-Based Rules - Definition and Classification

3.2.2. Resource-Based Rules

Also, we employ several resource-based rules which measure the importance of an activity in terms of either its resource demand, the scarcity of the resources used, or a combination of both. High priorities are usually assigned to potential resource-bottleneck activities which require large resource amounts or request scarce resources, the intention being to process the important activities as soon as possible to prevent less important activities from blocking them. We use five simple rules which assign priority values according to the *total resource* or the *dynamic relative demand* (TRD, DRD), the *total* or *dynamic resource scarcity* (TRS, DRS), or the *dynamic remaining capacity* (DRC) over all resources[6]. The resource-based measures are defined in columns 2 and 3 of Table 5. Note that measures are taken to equal zero whenever their divisor is zero or undefined.

[6] TRD is adapted from GRES in Kurtulus, Davis (1982) for the case of renewable resources, originated as GRD in Davis, Patterson (1975) for the case of nonrenewable resources. DRD, DRS, and DRC are adapted from RRU, RRS, and TRC in Schirmer (1993).

Extremum	Measure	Definition	Static vs. Dynamic	Local vs. Global	Simple vs. Composite	Serial vs. Parallel
MIN	TRD_j	$\leftarrow \Sigma_r k_{jr}$	S	L	S	S, P
MIN	DRD_j	$\leftarrow \Sigma_r k_{jr} / \max \{k_{j'r} \mid j' \in D_n\}$	D	G	S	S, P
MIN	TRS_j	$\leftarrow \Sigma_r k_{jr} / K_r$	S	L	S	S, P
MIN	DRS_j	$\leftarrow \Sigma_r k_{jr} / RK_{rn}$	D	L	S	P
MAX	DRC_j	$\leftarrow \Sigma_r (RK_{rn} - k_{jr})$	D	L	S	P

Table 5: Resource-Based Rules - Definition and Classification

4. Parameterized Random Sampling Methods

Deterministic heuristics return one sole solution for an instance, even if applied several times. Considering that this solution may be arbitrarily bad, determinism seems to be a major deficiency for heuristic methods. Therefore, *random sampling* methods proceed randomly, thus producing several solutions rather than only one per algorithm. Still, looking for a good solution by building a sample of purely random ones is rather inefficient: the majority of solutions found will be closer to the mean than to the maximum (or minimum) of the sample and thus fail to improve the best incumbent solution. Yet, even at the advent of such algorithms, then called random walk or Monte Carlo methods, King (1953) noted that biasing the probabilities may significantly improve sampling efficiency.

Hence, *biased random sampling* methods for scheduling, while still performing the selection process randomly, resolve conflicts according to probabilities which are proportional to priority values; in other words, the selection probabilities are biased by the priorities. Thus, in each scheduling step any eligible job *may* be chosen but those sharing higher priorities will have a higher probability of being selected. Note that tie-breaking rules are obsolete since ties cannot occur. Evidence gathered in several computational studies confirms the above expectation that such methods outperform traditional, deterministic approaches (Cooper 1976; Hart, Shogan 1987; Laguna et al. 1994; Drexl, Grünewald 1993). So, in addition to scheduling schemes and priority rules (cf. Section 3), we will use randomization (Rust 1997) to determine activity selection probabilities by means of *random sampling schemes.*

4.1. Classical Random Sampling Schemes

The idea of randomization in selecting between several alternative candidates is based on measures of attractiveness which are mapped monotonically into selection probabilities. Thus, a randomized method chooses among the available candidates according to probability values

which are biased to favour apparently attractive selections[7]. In the framework considered here, these measures are determined by priority rules. A random sampling scheme consists of a mapping p: $D_n \rightarrow [0,1]$ assigning a probability value p(j) to each candidate in the decision set. In essence, this mapping simply transforms the priority value of each candidate into a probability value. Of course, all probabilities sum to unity. In order to formalize the (randomized) selection once the probabilities are calculated, let denote $\vec{P}(D_n) = (p(\pi(1)),...,p(\pi(|D_n|)))$ the sequence of probability values of all candidates in a decision set D_n; w.l.o.g. we assume $\vec{P}(D_n)$ to be ordered by ascending activity index. Also, let denote $\zeta \in [0,1]$ a random number. Then, the activity j* to be selected is determined as

$$j^* \leftarrow \min \{ j \in D_n \mid j = \pi(k) \wedge \zeta \leq \sum_{i=1}^{k} p(\pi(i)) \} \tag{8}$$

In the sequel, we describe several random sampling schemes that have already been discussed in the literature.

4.1.1. Pure Random Sampling (RAS)

The most simple random sampling scheme consists of selecting among the available candidates in a purely random fashion. This, of course, renders the calculation of priority values obsolete. As it is usually (and hopefully) possible to find schemes doing better than this, RAS will only be used to provide a case against which the other schemes can be judged.

4.1.2. Biased Random Sampling (BRS)

The first such sampling scheme (BRS-C) was introduced by Cooper (1976) who calculates the probability of a candidate being selected by dividing its priority value by the sum of the priority values of all candidates in the decision set. However, in the case of extr = min, where those candidates having the smallest priorities should be accorded the highest selection probabilities, the priorities are modified beforehand into their reciprocals. To summarize,

$$v'(j) \leftarrow \begin{cases} 1/v(j) & \text{iff extr = min} \\ v(j) & \text{iff extr = max} \end{cases} \qquad (j \in D_n) \tag{9}$$

Afterwards, the selection probabilities are derived from these according to

---

[7] Cp. Rochat, Taillard (1995, p. 150). A simpler approach to randomization is used e.g. in the GRASP method, first proposed in Feo, Resende (1989), where the selection among attractive choices is done in a purely random fashion. For applications cf. Laguna et al. (1994) and the references mentioned therein.

$$p(j) \leftarrow \frac{v'(j)}{\sum_{j' \in D_n} v'(j')} \qquad (j \in D_n;\ v(j) > 0) \quad (10)$$

Alvarez-Valdés et al. (1989) proposed a modification of this scheme (BRS-AVT) where - in the case of extr = min - the priority values are modified differently by

$$v'(j) \leftarrow \begin{cases} M - v(j) & \text{iff extr} = \min \\ v(j) & \text{iff extr} = \max \end{cases} \qquad (j \in D_n) \quad (11)$$

M has to be large enough to guarantee that all modified priorities are nonnegative. Clearly, this scheme is applicable also if some priority values equal zero. In their study, for the critical-path-based priority rules with extr = min for example they used M ← T (Kolisch 1995, p. 99). Afterwards, the selection probabilities are derived from these according to (10).

4.1.3. Regret-Based Biased Random Sampling (RBRS)

Drexl introduced the concept of regrets, well-known from the field of decision theory[8], to the realm of random sampling (Drexl 1991; Drexl, Grünewald 1993). Here, the regret value of a candidate j measures the worst-case consequence that might possibly result from selecting another candidate. Let denote $V(D_n)$ the set of priority values of all candidates in a decision set D_n. Then, the regrets are computed as

$$v'(j) \leftarrow \begin{cases} \max V(D_n) - v(j) & \text{iff extr} = \min \\ v(j) - \min V(D_n) & \text{iff extr} = \max \end{cases} \qquad (j \in D_n) \quad (12)$$

and modified by

$$v''(j) \leftarrow (v'(j) + \varepsilon)^{\alpha} \qquad (j \in D_n) \quad (13)$$

where $\varepsilon \in R_{>0}$ and $\alpha \in R_{\geq 0}$. Having determined these values, the selection probabilities are derived from

$$p(j) \leftarrow \frac{v''(j)}{\sum_{j' \in D_n} v''(j')} \qquad (j \in D_n) \quad (14)$$

ε guarantees v''(j) to be nonzero; otherwise those candidates with priorities of zero could never be selected, an undesirable consequence in the presence of scarce resources. α allows to diminish or enforce the differences between the modified priorities for $\alpha<1$ or $\alpha>1$, respec-

---

8 Regrets underlie e.g. the Savage-Niehans and the Hurwicz rules for decision under uncertainty.

tively. Note that for $\alpha \rightarrow 0$ the selection process becomes pure random sampling, since all candidates will share the same probability of being selected. On the other extreme, for $\alpha = \infty$ the process behaves deterministic: since with increasing α the difference between the highest and the second-highest modified priority increases, the probability of the highest-prioritized candidate being selected converges to one.

4.2. New Random Sampling Schemes

In this section, we propose two new random sampling schemes. Motivation for doing so stems from certain critique on the classical schemes which is discussed before the schemes are introduced.

4.2.1. Motivation

Unfortunately, there are deficiencies associated with each of the above schemes. For both BRS schemes, these appear only if extr = min where the highest selection probabilities shall be assigned to the candidates with the smallest priorities. To achieve this, any transformation from the priorities to the probabilities has to rely either on division or subtraction. The former approach (cf. (9)) has been followed by Cooper (1976). The drawback to his BRS-C is that it is applicable only to priority rules which either have extr = max or which produce strictly nonzero priorities.

The latter approach (cf. (11)), which has been used more widely and recently (Alvarez-Valdés et al. 1989; Salewski et al. 1997; Böttcher et al. 1996), transforms the priorities by v'(j) $\leftarrow$ M - v(j) where M is large enough to keep all modified priorities nonnegative. However, this definition allows M to take arbitrarily large values which may distort the resulting probabilities. Consider a decision set of three selection candidates A, B, and C where the sequence $\vec{V}(D_n)$ of priority values (defined in the same way as $\vec{P}(D_n)$) is $\vec{V}(D_n) = (1, 2, 7)$. Figure 1 displays the probabilities resulting from different choices of M, assuming that no further modifications take place.

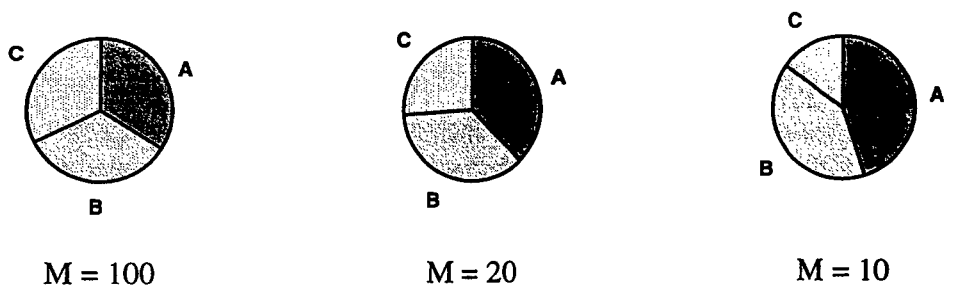


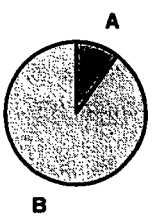
Figure 1: BRS-AVT - Effect of M on Selection Probabilities

In effect, values of M which are relatively large compared to the priorities reduce the influence the priority rules exert by biasing the sampling process and gear it in the direction of a purely random one.

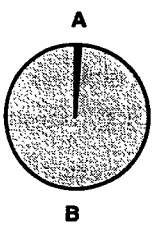
A different effect may appear for the RBRS scheme. Recall that here ε serves to keep the modified priorities nonzero; to achieve this, any small positive value would suffice. Yet, by definition ε may take arbitrary positive values, and for the sake of simplicity, most studies use ε = 1 (Drexl, Grünewald 1993; Kolisch, Drexl 1996, 1997; Kolisch 1996b; Salewski et al. 1997). But a fixed choice of ε may severely distort the resulting selection probabilities. Consider a situation where extr = max and the priority values of two candidates A and B are either $\bar{V}(D_n) = (1, 9)$ or (10, 90); for simplicity we let $\alpha = 1$. Since only the relation between the priority values should affect the selection probabilities, one would expect to see a probability for A of 10% as opposed to 90% for B, in both cases. Yet, when using ε = 1, the probability for A is 10% in the former only but 1% in the latter case, as shown in Figure 2.

ε	j	v	v'	v''	p
1	A	1	0	1	10%
	B	9	8	9	90%

ε	j	v	v'	v''	p
1	A	10	0	1	1%
	B	90	80	81	99%



$\bar{V}(D_n) = (1, 9)$



$\bar{V}(D_n) = (10, 90)$

Figure 2: RBRS - Effect of ε on Selection Probabilities (Varying Priorities, Constant ε)

Another way to look at this effect is to keep the priorities constant and regard the effect of different ε-values on the resulting probabilities. Consider the example depicted in Figure 3 where for three candidates A, B, and C with priorities $\bar{V}(D_n) = (0, 1, 1)$ and extr = min the probabilities derived are as shown for ε = 0.1 and ε = 1.

ε	j	v	v'	v''	p
0.1	A	0	1	1.1	85%
	B	1	0	0.1	8%
	C	1	0	0.1	8%

ε	j	v	v'	v''	p
1	A	0	1	2	50%
	B	1	0	1	25%
	C	1	0	1	25%

ε = 0.1 ε = 1

Figure 3: RBRS - Effect of ε on Selection Probabilities (Constant Priorities, Varying ε)

To summarize, whenever the values of ε are large relative to the priorities, they diminish the bias applied by the priority values and thus steer the sampling process towards pure random sampling. This contradicts the statement of Drexl, Grünewald (1993) that the choice of ε "is uncrucial for the behaviour of the algorithm".

4.2.2. Normalized Biased Random Sampling (NBRS)

We propose another approach which we term normalized biased random sampling (NBRS). In the case of extr = min the priorities v(j) are transformed, i.e.

$$v'(j) \leftarrow \begin{cases} \max V(D_n) - v(j) + \min V(D_n) & \text{iff extr} = \min \\ v(j) & \text{iff extr} = \max \end{cases} \qquad (j \in D_n) \quad (15)$$

Now, let denote $V'(D_n)^+$ the set of all positive transformed priorities of the candidates in D_n, i.e.

$$V'(D_n)^+ \leftarrow \{v'(j) \mid j \in D_n \wedge v'(j) > 0\} \qquad (16)$$

To ensure that each candidate is accorded a selection probability greater than zero, the priorities are then modified by

$$v''(j) \leftarrow \begin{cases} (v'(j)+\varepsilon)^{\alpha} & \text{iff} (\exists j \in D_n)\ v'(j) = 0 \\ (v'(j))^{\alpha} & \text{otherwise} \end{cases} \qquad (j \in D_n) \quad (17)$$

Here, ε is determined from

$$\varepsilon \leftarrow \begin{cases} \min V'(D_n)^+ / 10 & \text{iff } (\exists j \in D_n)\ v'(j) > 0 \\ 1 & \text{otherwise} \end{cases} \tag{18}$$

From the so-transformed priorities, the selection probabilities are derived as in (14). ε is applied only where necessary; still, the scheme ensures that no candidate is excluded from the selection process. Its first branch in (18) tends to keep its influence small whenever some transformed priorities are zero and some are not. The second branch applies only if all candidates in the decision set have zero priorities such that ε would be undefined; since then all priorities are identical, adding an arbitrary constant will give all candidates the same probability of being selected.

The naming of the scheme refers to the fact that (15) works to normalize the modified priorities v''(j) to the same interval as the original priorities v(j), such that the same probabilities are accorded to the most- and the least-preferred candidates, regardless of whether extr is max or min (cp. Figure 5).

extr	j	v	v'	v''	p
max	A	1	1	1	10%
	B	9	9	9	90%

extr	j	v	v'	v''	p
min	A	1	9	9	90%
	B	9	1	1	10%

extr = max extr = min

Figure 4: NBRS - Effect of extr on Selection Probabilities

4.2.3. Modified Regret-Based Biased Random Sampling (MRBRS)

This scheme is a variant of the RBRS, in that it computes regrets from the original priorities according to (12) and modifies them according to (13). However, rather than using a constant value of ε, ε is determined dynamically from

$$\varepsilon \leftarrow \begin{cases} \min V'(D_n)^+ / \delta & \text{iff } (\exists j \in D_n)\, v'(j) > 0 \\ 1 & \text{otherwise} \end{cases} \tag{19}$$

where δ is a positive integer. Selection probabilities are again derived from (14). Thus, the above actually defines a family MRBRS/δ of sampling schemes. We should add that while (19) could in principle also be used for the NBRS, our experimentation indicated that the results would not be altered significantly.

4.3. A Note on the Rules MIN LST and MIN SLK

It is well-known that under the PSS the rules MIN LST and MIN SLK are equivalent, in the sense of selecting the same candidates, because $EFST_j$ equals t_n for each $j \in D_n$ such that LST_j and SLK_j always differ by a constant amount t_n. This result was first proven by Davis, Patterson (1975); yet the proof is valid for deterministic scheduling only and does not carry over to all random sampling schemes alike[9].

Indeed, the equivalence of both rules holds true for the RBRS and all variants of the MRBRS but fails to hold for the BRS-C, the BRS-AVT, and the NBRS.

Theorem 1 Under the PSS, the priority rules MIN LST and MIN SLK are not equivalent for the BRS-C, the BRS-AVT, and the NBRS.

Proof: We give a simple counterexample with two activities 1 and 2 where $t_n = 4$ (cf. Table 6). We assume w.l.o.g. B = 10 for the BRS-AVT and $\alpha = 1$ for the NBRS.

RSS	j	v_{LST}	v'_{LST}	p_{LST}	v_{SLK}	v'_{SLK}	p_{SLK}
BRS-C	1	7	1/7	41.7%	3	1/3	25.0%
	2	5	1/5	58.3%	1	1	75.0%
BRS-AVT	1	7	3	37.5%	3	7	43.8%
	2	5	5	62.5%	1	9	56.3%
NBRS	1	7	5	41.7%	3	1	25.0%
	2	5	7	58.3%	1	3	75.0%

Table 6: A Counterexample on the Equivalence of MIN LST and MIN SLK

Clearly the probabilities differ between both rules for all three sampling schemes. ■

Theorem 2 Under the PSS, the priority rules MIN LST and MIN SLK are equivalent for the RBRS as well as all variants of the MRBRS/δ.

---

[9] The note in Kolisch (1995, p. 85) that both rules are generally equivalent is based upon the assumption that they always derive identical priorities; from the above, however, this assumption has to be refuted.

Proof: Since for all $j \in D_n$ alike the priorities $v_{LST}(j)$ and $v_{SLK}(j)$ differ by a constant, this constant affects their maximum max $V(D_n)$ in the same way: As $v_{SLK}(j) = v_{LST}(j) - t_n$, clearly max $V_{SLK}(D_n) = \max V_{LST}(D_n) - t_n$ and therefore $v'_{SLK}(j) = \max V_{SLK}(D_n) - v_{SLK}(j) = (\max V_{LST}(D_n) - t_n) - (v_{SLK}(j) - t_n) = \max V_{LST}(D_n) - v_{SLK}(j)$. Thus, the transformation (12) used in the RBRS as well as the MRBRS/δ renders transformed priorities which are identical for both rules. ■

5. Experimental Analysis

We start off by outlining the design used in our experimentation; in the remainder of this paper we present a comprehensive analysis of the algorithms described in the previous sections. To generate a valid set of comparisons between the algorithms considered, we validated the results obtained by comparing them to several other studies. We will juxtapose our findings with those reported in the literature where appropriate.

5.1. Experimental Design

The factors examined in our experiments are scheduling scheme (SS), random sampling scheme (RSS), priority rule, control parameter α, and number of iterations. Specifying a set of values for each factor describes over which levels it is varied during an *experiment*, while one value for each factor determines a *run* of an experiment. One such run may consist of one or several iterations per combination of algorithm and instance; for each such combination, the outcome of a run is reported in terms of both effectiveness and efficiency. Effectiveness is determined by considering the best schedule found for the instance in that run and measuring its deviation from the optimum as a percentage; efficiency captures the CPU-time required for the run, measured in terms of seconds. Measurements were taken using an implementation in Pascal, running on a P133 PC with 24 MB RAM under MS DOS 6.0.

As a test bed, we used the KSD-instance set J30 generated with ProGen (Kolisch, Sprecher 1997). Each instance is made up of 30 non-dummy activities. Each activity has a nonpreemptable duration of between one and ten periods and may require one or several of four renewable resources present. The number of successors and predecessors w.r.t. the precedence order varies between one and three for each activity. Systematically varied design parameters for these instances are the network complexity (NC), the resource factor (RF), and the resource strength (RS). NC is defined as the average number of non-redundant arcs per activity, RF determines the number of resources that are requested by each activity, and RS expresses resource scarcity measured between minimum and maximum demand. A more comprehensive characterization is given in Kolisch et al. (1995). The respective parameter levels used are shown in Table 7.

Parameter	Levels			
NC	1.5	1.8	2.1	
RS	0.2	0.5	0.7	1.0
RF	0.25	0.5	0.75	1.0

Table 7: KSD-Instance Set J30 - Varied Design Parameters

For each combination of these parameters, J30 contains ten instances, for a total of 480 instances. Of these, the 120 instances where RS = 1.0 are trivially solvable by the MPM-schedule. Of the remaining nontrivial sample, the exact algorithm of Demeulemeester, Herroelen (1992) found and verified the optimal solutions for 308 instances within a time limit of 3600 seconds per instance, taking on average 615.1 seconds per instance on a 386SX/15 PC; ten more instances were optimally solved but not verified within the time limit. These 308 instances have also been considered in several other studies (Mingozzi et al. 1994; Demeulemeester, Herroelen 1995; Naphade et al. 1995; Kolisch 1996a, b; Kolisch, Drexl 1996). The remaining 42 instances were only recently solved to optimality by a modified reimplementation of the above exact algorithm (Demeulemeester, Herroelen 1995), which on average improved the original solutions by 3.65 periods, requiring on average 29 seconds and at most 7200 seconds over all instances on an 486/25 PC. Embracing also these solutions, our work is the first analysis of priority rule-based heuristics covering the full range of instances comprised in J30.

For the most part, our experimentation used a full factorial design on the above factors, though excluding the BRS-AVT; the corresponding analysis is based upon the results of applying 502 different algorithms to the RCPSP. Several experiments of limited scope that were conducted in addition will be covered within the subsequent text, where appropriate.

5.2. *Computational Results*

In the sequel, we present the results of our computational experiments. We should emphasize that the effectiveness of algorithms will appear slightly lower than in other studies since our study includes the 52 rather hard instances unsolved at the time of earlier experimentation. Thus, our deviations from optimum are ceteris paribus larger than those reported elsewhere. Note that, unless stated otherwise, all results reported were obtained from performing 100 iterations of the respective algorithm.

5.2.1. *Effect Of Alpha*

For parameterized sampling schemes, the question of which control parameter to use for which scheme is paramount. Averaging the results of the full-factorial experiment over all levels of SS and α creates the impression that using $\alpha = 1$ does best, which would be in line with a similar finding for the RBRS by Kolisch (1996b). Yet, in many cases other choices of α produce better schedules. To demonstrate this, we have to disaggregate the results for the scheduling schemes and the priority rules employed. Table 8 presents the detailed results for all algorithms using the SSS. Results of the MRBRS/1000 are not included since they are strictly dominated by each of the other MRBRS-variants.

RSS	Alpha	GRPW	LFT	LST	SLK	MTS	SPT	WRUP	DRD	TRD	TRS
NBRS	1	3.25%	3.58%	3.46%	3.37%	2.62%	4.20%	3.40%	4.21%	4.40%	4.34%
	2	3.47%	3.27%	3.28%	3.15%	2.29%	5.06%	3.44%	5.03%	5.50%	5.30%
	3	3.77%	3.14%	3.01%	3.29%	2.24%	5.43%	3.71%	5.86%	6.00%	6.14%
RBRS	1	3.76%	2.07%	2.07%	3.34%	2.34%	4.92%	3.42%	4.18%	6.41%	4.03%
	2	4.70%	2.02%	1.97%	3.71%	2.32%	7.42%	3.45%	5.23%	9.93%	4.75%
	3	5.47%	2.21%	2.08%	3.83%	2.40%	9.26%	3.60%	6.31%	12.42%	5.24%
MRBRS/1	1	3.37%	2.21%	2.18%	3.15%	2.34%	4.47%	3.59%	4.67%	4.67%	4.91%
	2	3.90%	2.03%	2.00%	3.18%	2.31%	5.90%	4.47%	6.49%	6.39%	6.72%
	3	4.67%	2.10%	1.99%	3.38%	2.36%	7.50%	5.38%	8.00%	8.30%	8.43%
MRBRS/10	1	4.24%	1.99%	1.95%	3.21%	2.31%	6.88%	4.45%	7.18%	7.31%	7.46%
	2	5.26%	2.19%	2.13%	3.69%	2.52%	10.66%	6.15%	11.12%	11.89%	11.72%
	3	6.34%	2.38%	2.17%	3.93%	2.84%	12.63%	7.63%	14.09%	14.76%	15.23%
MRBRS/100	1	4.81%	2.17%	1.89%	3.57%	2.52%	9.74%	5.17%	10.01%	10.66%	10.71%
	2	5.87%	2.24%	2.13%	3.82%	2.64%	12.53%	6.73%	13.54%	14.79%	14.79%
	3	6.47%	2.46%	2.22%	4.00%	2.89%	13.48%	7.81%	14.78%	15.78%	16.15%

Table 8: Effect of α - Deviations (SSS)

RSS	Alpha	GRPW	IRSM	LFT	LST	SLK	MTS	RSM	SPT	WCS	WRUP
NBRS	1	3.17%	2.65%	3.16%	3.18%	3.22%	2.71%	2.62%	3.29%	3.17%	3.15%
	2	3.21%	2.68%	3.18%	3.16%	3.09%	2.47%	2.73%	3.96%	3.05%	2.92%
	3	3.55%	2.68%	3.06%	3.12%	2.94%	2.49%	2.66%	4.31%	3.05%	3.03%
RBRS	1	3.48%	2.45%	2.40%	2.50%	2.50%	2.58%	2.50%	3.88%	2.36%	3.17%
	2	4.24%	2.57%	2.61%	2.72%	2.72%	2.63%	2.65%	4.95%	2.56%	3.02%
	3	5.18%	2.73%	2.87%	3.11%	3.11%	2.81%	2.90%	6.18%	2.81%	3.05%
MRBRS/1	1	3.19%	2.70%	2.64%	2.70%	2.70%	2.68%	2.75%	3.54%	2.57%	3.06%
	2	3.49%	2.49%	2.46%	2.48%	2.48%	2.50%	2.49%	3.97%	2.43%	3.24%
	3	3.94%	2.52%	2.46%	2.49%	2.49%	2.53%	2.55%	4.82%	2.37%	3.54%
MRBRS/10	1	3.69%	2.49%	2.55%	2.51%	2.51%	2.61%	2.53%	4.81%	2.42%	3.45%
	2	5.01%	2.88%	2.83%	3.00%	3.00%	3.10%	2.95%	6.88%	2.78%	4.47%
	3	6.02%	3.07%	3.16%	3.37%	3.37%	3.45%	3.22%	8.72%	2.99%	5.63%

Table 9a: Effect of α - Deviations (PSS, Network- and Time-Based Rules)

Underlined entries denote, for each combination of priority rule and random sampling scheme, the lowest deviation over all α-values tested, thus allowing to look up the best α-value for each such combination. The double-underlined entry is the minimum of all underlined entries. Except of the MRBRS/10 (as well as MRBRS/100 and MRBRS/1000), where $\alpha = 1$ consistently performs best, the general picture is inconclusive. Hence, no general recommendation on the choice of α can be given.

RSS	Alpha	DRC	DRD	DRS	TRD	TRS
NBRS	1	3.58%	3.70%	3.70%	3.64%	3.67%
	2	3.83%	4.24%	4.10%	4.05%	4.02%
	3	4.12%	4.51%	4.49%	4.70%	4.69%
RBRS	1	5.04%	3.74%	3.50%	5.04%	3.61%
	2	7.54%	4.41%	4.14%	7.54%	3.94%
	3	9.15%	5.09%	4.69%	9.15%	4.36%
MRBRS/1	1	4.06%	3.94%	3.91%	4.06%	4.00%
	2	4.75%	4.67%	4.94%	4.75%	4.92%
	3	5.70%	5.61%	5.92%	5.70%	5.93%
MRBRS/10	1	5.77%	5.63%	5.91%	5.77%	6.05%
	2	8.69%	8.52%	8.85%	8.69%	9.08%
	3	11.10%	11.02%	11.26%	11.10%	11.41%

Table 9b: Effect of α - Deviations (PSS, Resource-Based Rules)

Table 9 details the situation for the PSS; again results for the (dominated) schemes MRBRS/100 and MRBRS/1000 were omitted. Here, the effect of α is more consistent: For the RBRS, $\alpha = 1$ gives the best results; this finding coincides with that of Kolisch (1996b). For the MRBRS/10 (as well as for MRBRS/100 and MRBRS/1000) $\alpha = 1$ outperform other choices. Only for the NBRS and the MRBRS/1, no general recommendation on the choice of α can be given.

RSS	*Averages*			*Percentages*		
	1	2	3	1	2	3
NBRS	0.51	0.84	0.85	100.0%	164.9%	166.1%
RBRS	0.46	0.76	0.77	100.0%	166.9%	168.3%
MRBRS/1	0.50	0.85	0.86	100.0%	169.1%	171.3%
MRBRS/10	0.52	0.90	0.90	100.0%	173.4%	173.7%
MRBRS/100	0.52	0.90	0.92	100.0%	172.9%	177.0%
MRBRS/1000	0.52	0.90	0.90	100.0%	172.9%	173.4%

Table 10: Effect of α - CPU Times

The effect of α on the efficiency can be taken from Table 10 where average CPU times are reported both as absolute numbers and percentages of the CPU times using $\alpha = 1$. We chose to

report the resulting numbers aggregated over the priority rules since a disaggregation, as done for effectivity in Tables 8 and 9, fails to reveal significant differences. Obviously, choosing an $\alpha \neq 1$ increases the computation times by two thirds, which is due to the additional effort required to determine the power of the transformed priorities. The reader should note that the exact amount of this increase may vary with different computer environments; yet additional experimentation on different implementations of the power function, which is not further detailed here, exposed no substantial variations.

5.2.2. *Effect Of Priority Rules*

We now consider the influence exerted by the priority rules. Again, we cover the results for the two scheduling schemes separately since the performance of some rules differs markedly between them[10]. In Table 11, the results pertaining to the SSS are summarized. To facilitate comparisons, we also give the results of the deterministic versions of the priority rules.

RSS	Alpha	GRPW	LFT	LST	SLK	MTS	SPT	WRUP	DRD	TRD	TRS
DETERM	N/A	13.75%	7.44%	6.56%	5.58%	8.74%	22.80%	15.64%	23.13%	23.38%	24.01%
BRS-C	N/A	3.25%	3.60%	3.51%	3.43%	2.62%	4.15%	3.40%	4.40%	4.32%	4.31%
NBRS	1	3.25%	3.58%	3.46%	3.37%	2.62%	4.20%	3.40%	4.21%	4.40%	4.34%
	2	3.47%	3.27%	3.28%	3.15%	2.29%	5.06%	3.44%	5.03%	5.50%	5.30%
	3	3.77%	3.14%	3.01%	3.29%	2.24%	5.43%	3.71%	5.86%	6.00%	6.14%
RBRS	1	3.76%	2.07%	2.07%	3.34%	2.34%	4.92%	3.42%	4.18%	6.41%	4.03%
	2	4.70%	2.02%	1.97%	3.71%	2.32%	7.42%	3.45%	5.23%	9.93%	4.75%
	3	5.47%	2.21%	2.08%	3.83%	2.40%	9.26%	3.60%	6.31%	12.42%	5.24%
MRBRS/1	1	3.37%	2.21%	2.18%	3.15%	2.34%	4.47%	3.59%	4.67%	4.67%	4.91%
	2	3.90%	2.03%	2.00%	3.18%	2.31%	5.90%	4.47%	6.49%	6.39%	6.72%
	3	4.67%	2.10%	1.99%	3.38%	2.36%	7.50%	5.38%	8.00%	8.30%	8.43%
MRBRS/10	1	4.24%	1.99%	1.95%	3.21%	2.31%	6.88%	4.45%	7.18%	7.31%	7.46%
	2	5.26%	2.19%	2.13%	3.69%	2.52%	10.66%	6.15%	11.12%	11.89%	11.72%
	3	6.34%	2.38%	2.17%	3.93%	2.84%	12.63%	7.63%	14.09%	14.76%	15.23%
MRBRS/100	1	4,81%	2,17%	1.89%	3,57%	2,52%	9,74%	5,17%	10.01%	10.66%	10.71%
	2	5,87%	2,24%	2.13%	3,82%	2,64%	12,53%	6,73%	13.54%	14.79%	14.79%
	3	6,47%	2,46%	2.22%	4,00%	2,89%	13,48%	7,81%	14.78%	15.78%	16.15%

Table 11: Effect of Priority Rules - Deviations (SSS)

Entries underlined are the respective row minima, reflecting the best priority rule for each combination of random sampling scheme and α-value. It is evident from the data that the particular choice of α bears no influence on the suitability of different priority rules, thus the underlined entries essentially indicate the best rules for each sampling scheme. For the BRS-C

[10] Although this has been observed for the rule SLK in Kolisch (1995, p. 111), there the effect of the rules is reported as an average over both scheduling schemes.

and NBRS, MTS is the bestperforming rule, for the other schemes it is LST. This is in line with the report of Kolisch (1996b) who found LST to be the best rule for the SSS.

RSS	Alpha	GRPW	IRSM	LFT	LST	SLK	MTS	RSM	SPT	WCS	WRUP
DETERM	N/A	10.47%	5.92%	5.86%	6.04%	6.04%	6.66%	6.36%	13.09%	5.17%	9.78%
BRS-C	N/A	3.17%	2.63%	3.15%	3.15%	3.20%	2.71%	2.64%	3.38%	3.15%	3.15%
NBRS	1	3.17%	2.65%	3.16%	3.18%	3.22%	2.71%	2.62%	3.29%	3.17%	3.15%
	2	3.21%	2.68%	3.18%	3.16%	3.09%	2.47%	2.73%	3.96%	3.05%	2.92%
	3	3.55%	2.68%	3.06%	3.12%	2.94%	2.49%	2.66%	4.31%	3.05%	3.03%
RBRS	1	3.48%	2.45%	2.40%	2.50%	2.50%	2.58%	2.50%	3.88%	2.36%	3.17%
	2	4.24%	2.57%	2.61%	2.72%	2.72%	2.63%	2.65%	4.95%	2.56%	3.02%
	3	5.18%	2.73%	2.87%	3.11%	3.11%	2.81%	2.90%	6.18%	2.81%	3.05%
MRBRS/1	1	3.19%	2.70%	2.64%	2.70%	2.70%	2.68%	2.75%	3.54%	2.57%	3.06%
	2	3.49%	2.49%	2.46%	2.48%	2.48%	2.50%	2.49%	3.97%	2.43%	3.24%
	3	3.94%	2.52%	2.46%	2.49%	2.49%	2.53%	2.55%	4.82%	2.37%	3.54%
MRBRS/10	1	3.69%	2.49%	2.55%	2.51%	2.51%	2.61%	2.53%	4.81%	2.42%	3.45%
	2	5.01%	2.88%	2.83%	3.00%	3.00%	3.10%	2.95%	6.88%	2.78%	4.47%
	3	6.02%	3.07%	3.16%	3.37%	3.37%	3.45%	3.22%	8.72%	2.99%	5.63%

Table 12a: Effect of Priority Rules - Deviations (PSS, Network- and Time-Based-Based Rules)

RSS	Alpha	DRC	DRD	DRS	TRD	TRS
DETERM	N/A	15.83%	15.93%	15.80%	15.83%	15.93%
BRS-C	N/A	3.58%	3.58%	3.64%	3.75%	3.62%
NBRS	1	3.58%	3.70%	3.70%	3.64%	3.67%
	2	3.83%	4.24%	4.10%	4.05%	4.02%
	3	4.12%	4.51%	4.49%	4.70%	4.69%
RBRS	1	5.04%	3.74%	3.50%	5.04%	3.61%
	2	7.54%	4.41%	4.14%	7.54%	3.94%
	3	9.15%	5.09%	4.69%	9.15%	4.36%
MRBRS/1	1	4.06%	3.94%	3.91%	4.06%	4.00%
	2	4.75%	4.67%	4.94%	4.75%	4.92%
	3	5.70%	5.61%	5.92%	5.70%	5.93%
MRBRS/10	1	5.77%	5.63%	5.91%	5.77%	6.05%
	2	8.69%	8.52%	8.85%	8.69%	9.08%
	3	11.10%	11.02%	11.26%	11.10%	11.41%

Table 12b: Effect of Priority Rules - Deviations (PSS, Resource-Based Rules)

The corresponding results for the PSS are comprised in Table 12. For all variants of the MRBRS as well as the RBRS, WCS is the best choice. Regarding the effectiveness of the resource-based rules, we may summarize that these rules perform substantially better under the

PSS than the SSS. Still, even the best combination of such rule and SS, viz. DRS under the PSS, is markedly outperformed by each of the promising rules.

Taking up this finding, we now propose to circle out the most promising rules for each scheduling scheme and accordingly restrict subsequent analyses. Motivation for excluding certain rules stems from the fact that aggregate results will be distorted by bad rules whenever some algorithmic components are more sensitive than others to the performance of individual rules. To identify such rules, which are to be excluded, we use two criteria, one being rule-independent, the other depending on the performance of the individual rules. The first criterion establishes that no rule strictly dominated by the RAS is promising: as RAS, due to its obviating the need to compute probabilities, is more efficient than any other randomized algorithm considered here, any algorithm exhibiting the same or worse effectiveness than RAS should be discontinued[11]. To keep the comparison on fair grounds, RAS should of course be run for the same number of iterations as the algorithms judged against it. The second criterion excludes all rules which are dominated by at least one other rule: they do not seem to offer any potential over their companions. Applying these criteria to our experimental setting, promising rules are those which perform best for at least one combination of SS, RSS, and α-value and outperform the RAS (whose results are provided in Table 22 below). Accordingly, we can identify from Table 11 the rules LFT, LST, and MTS as promising for the SSS. Following these criteria, one may also characterize IRSM, LFT, MTS, and RSM as promising for the PSS. Still, we will also accept SLK and LST since their effectiveness is in most cases close, for some instance clusters even better than that of the above rules. Following such an argument may prove valuable especially when attempting to identify suitable candidates for a multi-priority rule approach (Boctor 1990; Khattab, Choobineh 1991) or composite priority rules (Haase et al. 1997). We assemble the promising rules in Table 13.

SS	Priority Rules						
SSS	LFT	LST	MTS				
PSS	IRSM	LFT	LST	MTS	SLK	RSM	WCS

Table 13: Promising Priority Rules

Table 14 demonstrates the effect of these rules on the CPU times. To exclude the effects incurred by the scheduling schemes, the presentation is divided among both schemes; to exclude

[11] In an analysis of deterministic scheduling algorithms, Lawrence (1985) has employed pure random selection as a means of resolving scheduling conflicts and used the results as a yardstick to separate those algorithms (priority rules in his case) minimizing the project makespan from those actually maximizing it.

effects from different α-values (cp. Table 10), α is fixed to one for all sampling schemes except for the BRS-C.

RSS	*SSS*			*PSS*						
	LFT	LST	MTS	IRSM	LFT	LST	SLK	MTS	RSM	WCS
BRS-C	0.42	0.42	0.39	0.48	0.36	0.36	0.36	0.35	0.44	0.46
NBRS	0.59	0.59	0.59	0.59	0.42	0.46	0.42	0.42	0.53	0.53
RBRS	0.52	0.52	0.51	0.52	0.40	0.40	0.41	0.40	0.48	0.52
MRBRS/1	0.58	0.57	0.57	0.55	0.44	0.44	0.43	0.44	0.51	0.54
MRBRS/10	0.60	0.60	0.60	0.57	0.45	0.45	0.45	0.45	0.53	0.56
MRBRS/100	0.61	0.60	0.60	0.57	0.45	0.45	0.45	0.45	0.53	0.56
MRBRS/1000	0.61	0.60	0.61	0.57	0.45	0.45	0.45	0.45	0.53	0.56

Table 14: Effect of Priority Rules - CPU Time

5.2.3. Effect Of Alpha - Revisited

Tables 8 and 9 did not proffer any general guidance on which α-values are most effective for all priority rules alike. Yet restricting the focus to the promising rules allows to deliver conclusive recommendations. Table 15 shows the best-performing α-value for each combination of RSS and SS. Note that these values produce the best results averaged over all promising rules; for certain combinations of RSS, SS, and priority rules the individually best α-value may be different, though (cp. Table 21).

RSS	SSS	PSS
NBRS	3	3
RBRS	2	1
MRBRS/1	2	2
MRBRS/10	1	1
MRBRS/100	1	1
MRBRS/1000	1	1

Table 15: Best α-Values

Following the lines of Drexl (1991), we have also experimented with non-integer values of α in order to further improve the performance of the algorithms considered, selecting values in the vicinity of the best-performing integer ones. Without providing the corresponding results let us just mention that in no case the effectiveness improved by more than 0.005%. Considering the fact that non-integer α-values increase the computation times required by more than 40%, it seems to be more worthwhile to restrict oneself to integer values and simply increase the number of iterations in order to obtain better solutions.

Recall that, by modifying the selection probabilities, α affects the bias that the priority values apply to the selection process: large values render the process more deterministic while small values make it more random. In much the same way, the divisor δ determining the particular variant of the MRBRS/δ affects the selection process: large values increase the bias, thus steering the selection process towards more determinism, whereas small values gear it towards more random. One is thus - correctly, we might add - led to expect different values of α to perform best for different values of δ. In order to demonstrate this interaction, we chart in Figure 5 the effectiveness of different variants of the MRBRS/δ where $\delta \in \{1, 10, 100, 1000\}$ and different values of α. Results reflect the promising rules only. As additional experimentation not reported here indicated that interpolating between the data points generated is viable, we chart them by way of surface diagrams.

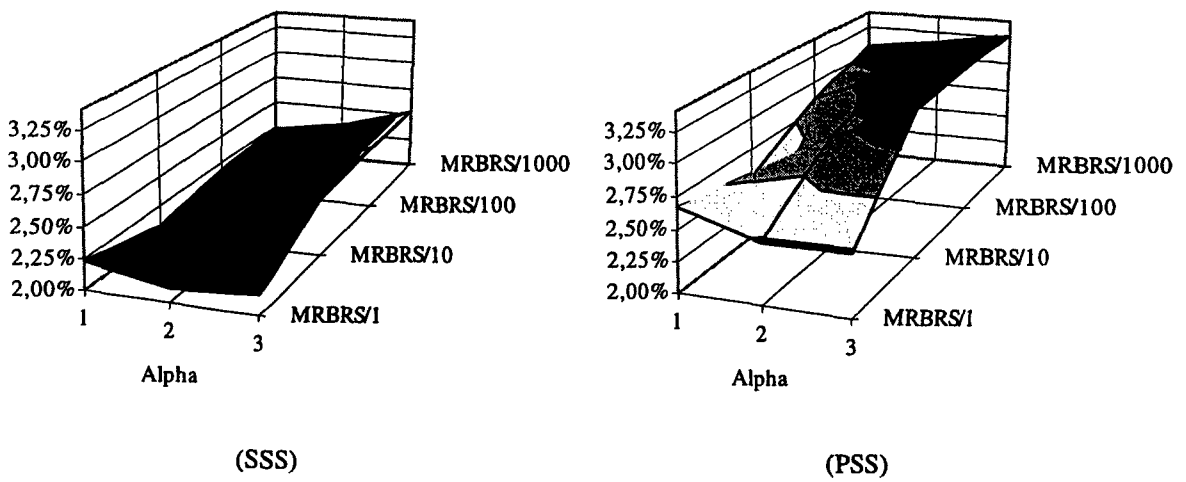


Figure 5: MRBRS - Effect of α - Deviations

Indeed, under the SSS for $\delta = 1$ $\alpha = 2$ performs best (deviation 2.11%) while for the other δ-values $\alpha = 1$ does best (deviation 2.08% or higher). Under the PSS, the picture is virtually the same: for $\delta = 1$ the best choice is $\alpha = 2$ (deviation 2.48%), for the other δ-values $\alpha = 1$ is dominant (deviation 2.52% or higher).

5.2.4. Effect of Instance Sets

As far as they pertain to the RBRS, some of the results discussed so far deviate somewhat from those reported in Kolisch (1996b). As it seems reasonable to expect this discrepancy to arise from the different instance sets used, we divide the analysis between the 308 instances used in his study and the 52 instances additionally included in our work. The corresponding results are summarized in Tables 16 and 17.

Priority Rule	*308 Instances*			*52 Instances*			*All Instances*		
	1	2	3	1	2	3	1	2	3
LFT	1,26%	1,23%	1,36%	6,92%	6,71%	7,29%	2,07%	2,02%	2,21%
LST	1,24%	1,16%	1,33%	7,00%	6,72%	6,56%	2,07%	1,97%	2,08%
MTS	1,54%	1,61%	1,67%	7,07%	6,53%	6,67%	2,34%	2,32%	2,40%

Table 16: Effect of Priority Rules and α - Deviations (SSS, RBRS)

For the serial variant of the RBRS, our above advice to use $\alpha = 2$ (cf. Table 15) differs with what Kolisch (1996b) recommends; he found $\alpha = 1$ to be most effective. The underlined row minima in Table 16 reflect the best value of α for each rule. Still, in each instance set $\alpha = 2$ remains the best choice, regardless of whether the average over all promising rules or only the best rule LST are considered.

In Table 17, underlined entries denote the best combination of priority rule and α-value for each instance set. For the parallel variant of the RBRS, Kolisch (1996b) found WCS with a setting of $\alpha = 1$ to outperform all other rules on the 308 instances attempted. This result could not be reproduced, even if only by a small margin. However, our results concur with his when either the complete instance set J30 is tackled or α-values greater than one are used.

Priority Rule	*308 Instances*			*52 Instances*			*All Instances*		
	1	2	3	1	2	3	1	2	3
IRSM	2,03%	2,15%	2,22%	4,94%	5,03%	5,71%	2,45%	2,57%	2,73%
LFT	2,00%	2,22%	2,48%	4,75%	4,87%	5,21%	2,40%	2,61%	2,87%
LST	2,12%	2,29%	2,71%	4,73%	5,26%	5,48%	2,50%	2,72%	3,11%
MSLK	2,12%	2,29%	2,71%	4,73%	5,26%	5,48%	2,50%	2,72%	3,11%
MTS	2,13%	2,29%	2,40%	5,24%	4,67%	5,25%	2,58%	2,63%	2,81%
RSM	2,01%	2,19%	2,33%	5,37%	5,40%	6,22%	2,50%	2,65%	2,90%
WCS	2,01%	2,20%	2,43%	4,41%	4,67%	5,02%	2,36%	2,56%	2,81%

Table 17: Effect of Priority Rules and α - Deviations (PSS, RBRS)

5.2.5. Effect Of Scheduling Schemes

Kolisch (1996b) was the first to note that the mutual dominance of serial and parallel scheduling depends, among other factors, on the number of iterations performed. To clarify this point, let us restrict ourselves to the promising rules LFT, LST, and MTS, which are the only ones applicable in both scheduling schemes. Table 18 shows the effectiveness of both schemes, using the MRBRS/10 with the best α-value from Table 15 as a vehicle.

Rule	SS	10	40	70	100
LFT	Parallel	3,98%	2,95%	2,67%	2,55%
	Serial	4,21%	2,62%	2,20%	1,99%
LST	Parallel	3,77%	2,91%	2,63%	2,51%
	Serial	4,02%	2,47%	2,16%	1,95%
MTS	Parallel	4,10%	3,03%	2,81%	2,61%
	Serial	4,56%	2,98%	2,56%	2,31%

Table 18: Effect of Scheduling Schemes and Iterations - Deviations

Clearly, the PSS is more effective on the short run whereas on the long run the SSS capitalizes better on the larger iteration numbers. We will also see below that for 100 iterations the SSS succeeds in solving to optimality a larger portion of the instances attempted (cf. Table 22). These observations are consistent with theoretical insight since the set of non-delay schedules searched by the PSS will not always contain an optimal solution whereas each optimum belongs to the larger set of active schedules which are searched by the SSS (Kolisch 1996b). Although our test instances do not allow such conclusions to be drawn, the iteration number at which the PSS is overtaken by the SSS might well be expected to vary with the size of the instances attempted.

We should add that the picture exposed is virtually the same over all RSS, which frees us from reproducing the complete numbers for the other schemes. In a few cases, the PSS is still dominant after 40 iterations; from 70 iterations on the SSS is always superior, though. Let us also add without providing numerical evidence the interesting observation that promising and bad rules react differently to increasing iteration numbers. Indeed, for the bad rules the PSS remains dominant even after 100 iterations, regardless of the RSS used.

The bearing of the SS on efficiency is shown in Table 19 where the CPU times for the SS-RSS combinations are averaged over all priority rules. In order to control for the influence of different α-values, again α was fixed to one where appropriate.

RSS	*Averages*		*Percentages*	
	Serial	Parallel	Serial	Parallel
BRS-C	0.40	0.35	113.6%	100.0%
NBRS	0.58	0.48	121.2%	100.0%
RBRS	0.49	0.45	110.8%	100.0%
MRBRS/1	0.56	0.48	117.0%	100.0%
MRBRS/10	0.57	0.49	115.8%	100.0%
MRBRS/100	0.58	0.49	116.8%	100.0%
MRBRS/1000	0.58	0.49	116.8%	100.0%

Table 19: Effect of Scheduling Schemes - CPU Times

The SSS takes between 11% and 21% longer than its counterpart, which is due to the fact that it has to constantly manage the RxT matrix of remaining capacities while the PSS has to consider only an R-vector of these values in each scheduling step (Kolisch 1995, p. 106).

5.2.6. Effect Of Random Sampling Schemes

We are now in a position to undertake a conclusive examination of the random sampling schemes tested. Limiting the presentation for each sampling scheme to the results of applying the promising rules with the best α-value of Table 15 allows to identify the most promising rules for each sampling scheme and thus the most promising algorithms. The respective information is given in Table 20. Recall from above that these α-values are not necessarily the individually best ones for each combination of RSS, SS, and rule. Indeed the deviation entry tagged by * (**) can be further reduced to 1.99% (2.37%) by using a setting of $\alpha = 3$. Again, the dominated MRBRS/1000 has been omitted.

RSS	*SSS*			*PSS*						
	LFT	LST	MTS	IRSM	LFT	LST	SLK	MTS	RSM	WCS
BRS-C	3.60%	3.51%	2.62%	2.63%	3.15%	3.15%	3.20%	2.71%	2.64%	3.15%
NBRS	3.14%	3.01%	2.24%	2.68%	3.06%	3.12%	2.94%	2.49%	2.66%	3.05%
RBRS	2.02%	1.97%	2.32%	2.45%	2.40%	2.50%	2.50%	2.58%	2.50%	2.36%
MRBRS/1	2.03%	2.00%*	2.31%	2.49%	2.46%	2.48%	2.48%	2.50%	2.49%	2.43%**
MRBRS/10	1.99%	1.95%	2.31%	2.49%	2.55%	2.51%	2.51%	2.61%	2.53%	2.42%
MRBRS/100	2.17%	1.89%	2.52%	2,78%	2,80%	2,94%	2,94%	3,07%	2,83%	2,71%

Table 20: Effect of Random Sampling Schemes - Deviations (Promising Algorithms)

Underlined are the column minima, indicating the best sampling scheme for each priority rule. Let us first discuss the case of serial scheduling. W.r.t. the sampling schemes, BRS-C is clearly inferior to all other schemes. For the rule MTS, the NBRS excels the other schemes but this deviation is clearly undercut by other algorithms. Discounting the rather poorly performing schemes BRS-C and NBRS, the rule LST strictly dominates the other rules. A particularly noteworthy finding is that the RBRS, so far the best known sampling scheme (Kolisch 1996b), is strictly dominated by the MRBRS/10. An even higher effectiveness for the SSS is demonstrated by the MRBRS/100 using LST; it improves the best performance of the RBRS by 4.1%. Yet, taking also its efficiency into account reveals it to do even better. Since the RBRS achieves its best performance with a setting of $\alpha = 3$, its computation time is about 50% higher than the one required by the MRBRS/100 which uses $\alpha = 1$ in its best configuration. Therefore, in an additional experiment, we invested this time to perform additional iterations, thus using the same total computation time allotted to RBRS. The resulting deviation of the MRBRS/100 reduces to 1.77%, or an improvement of 10.20% over the RBRS.

Under the parallel scheduling scheme, the priority rules LFT, WCS, and IRSM as a group outperform the remaining rules. Considering the sampling schemes, except of the combination (NBRS, MTS) whose deviation is practically identical to that of (MRBRS/1, MTS), again BRS-C and NBRS consistently perform worse than the other three schemes. At first glance a surprising result is that for the PSS, the BRS-C is not much worse than the other RSS considered. This conforms, however, with theoretical insight: As parallel algorithms search the smaller solution space, it is straightforward that the deviations between different parallel algorithms should be smaller than those between different serial ones, especially so with larger iteration numbers. Of the other three sampling schemes, none succeeds to strictly dominate the other two. All in all, the MRBRS/1 and RBRS can be considered the best schemes for parallel scheduling. Deciding between them is a close call, however, with the MRBRS/1 clearly dominating the RBRS for four of the seven promising rules and the RBRS - using either WCS or LFT - producing the best schedules of all parallel algorithms examined. In other words, the MRBRS/1 should be selected if all promising rules, the RBRS if only the best algorithm shall be employed.

From these results, the best individual priority rule and α-value for each combination of scheduling and sampling scheme can be extracted as listed in Table 21.

RSS	*SSS*		*PSS*	
	Rule	Alpha	Rule	Alpha
DETERM	SLK	N/A	IRSM	N/A
BRS-C	MTS	N/A	IRSM	N/A
NBRS	MTS	3	MTS	3
RBRS	LST	2	WCS	1
MRBRS/1	LST	3	WCS	3
MRBRS/10	LST	1	WCS	1
MRBRS/100	LST	1	WCS	1
MRBRS/1000	LST	1	WCS	1

Table 21: Best Rules and α-Values

Another glance on the effectiveness of the best algorithms for each SS and RSS is provided in Table 22, listing for each combination the results obtained from employing the individually best rule and α-value, respectively. Avg (SD, Max) denotes the average (standard deviation, maximum) of the deviations from optimum, % Opt the percentage of instances optimally solved.

RSS	*SSS*				*PSS*			
	Avg	SD	Max	% Opt	Avg	SD	Max	% Opt
DETERM	5.58%	6.26%	27.38%	38.61%	5.17%	5.07%	26.39%	26.67%
RAS	3.70%	4.40%	17.65%	44.72%	3.31%	3.41%	15.46%	33.89%
BRS-C	2.62%	3.38%	14.29%	50.83%	2.71%	2.89%	13.64%	37.22%
NBRS	2.24%	3.03%	12.90%	53.61%	2.49%	2.78%	13.64%	38.61%
RBRS	1.97%	3.09%	16.13%	59.44%	2.36%	2.67%	13.64%	40.28%
MRBRS/1	1.99%	3.12%	14.44%	58.61%	2.37%	2.69%	13.64%	40.83%
MRBRS/10	1.95%	3.07%	13.89%	59.44%	2.42%	2.73%	13.64%	40.56%
MRBRS/100	1.89%	2.92%	11.90%	58.61%	2.71%	2.97%	17.78%	37.50%
MRBRS/1000	1.97%	2.99%	12.90%	58.89%	2.91%	3.16%	17.78%	36.39%

Table 22: Effect of Random Sampling Schemes - Summary (Best Algorithms)

5.2.7. Effect Of Iterations

Let us now turn our attention to the influence of iterations. Apart from affecting the suitability of scheduling schemes, they also bear on the effectiveness of the different RSS. Consider Table 23 which is based upon employing the best combination of priority rule and α-value for each combination of RSS and SS (recall that these α-values may differ from those of Table 15 which covers all promising rules rather than only the best one). A graphical summary, including only the respective best-performing MRBRS-variant, is given in Figure 6.

RSS	*Serial*						*Parallel*					
	Rule	Alpha	10	40	70	100	Rule	Alpha	10	40	70	100
BRS-C	MTS	N/A	5,64%	3,49%	2,96%	2,62%	IRSM	N/A	4,18%	3,08%	2,79%	2,63%
NBRS	MTS	3	4,23%	2,93%	2,51%	2,24%	MTS	2	3,95%	2,82%	2,62%	2,47%
RBRS	LST	2	3,76%	2,42%	2,15%	1,97%	WCS	1	3,85%	3,00%	2,79%	2,36%
MRBRS/1	LST	3	3,60%	2,43%	2,08%	1,99%	WCS	3	3,51%	2,69%	2,49%	2,37%
MRBRS/10	LST	1	4,02%	2,47%	2,16%	1,95%	WCS	1	3,82%	2,79%	2,53%	2,42%
MRBRS/100	LST	1	3,75%	2,42%	2,11%	1,89%	WCS	1	3,79%	3,08%	2,80%	2,71%
MRBRS/1000	LST	1	3,73%	2,50%	2,21%	1,97%	WCS	1	3,83%	3,15%	3,00%	2,91%

Table 23: Effect of Iterations and Random Sampling Schemes - Deviations

The underlined column minima indicate the best-performing sampling scheme, respectively. It is evident that the relative effectiveness of the sampling scheme may depend on changes in the number of iterations performed. Except for the PSS using 100 iterations, the MRBRS/δ with δ = 1 or δ = 100 dominates all other RSS, including the RBRS, although the specific choice of the best δ-value is affected by the number of iterations performed. Under the SSS, even the NBRS performs better than the RBRS.

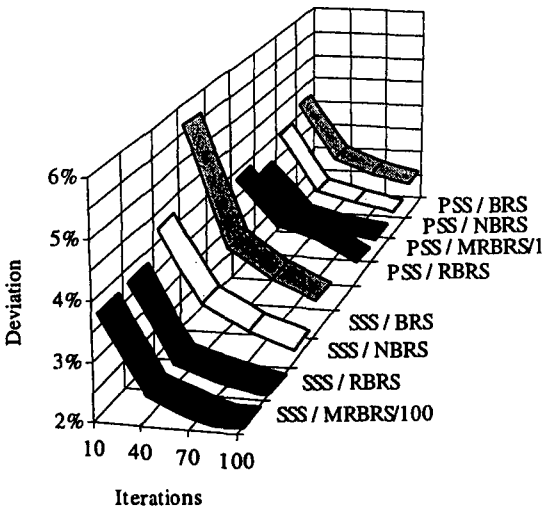


Figure 6: Effect of Iterations and Random Sampling Schemes - Deviations

Finally, we devote ourselves to the question of how much the deviations can be decreased by increasing iterations. We therefore conducted another experiment in which several RSS were employed under both SS, using the promising rules only with $\alpha = 1$, and running these for 500 iterations. In Figure 7, we summarize the results for the MRBRS/10. The results are representative of those for the RBRS and other MRBRS-variants. For comparative purposes, we also include the results of RAS.

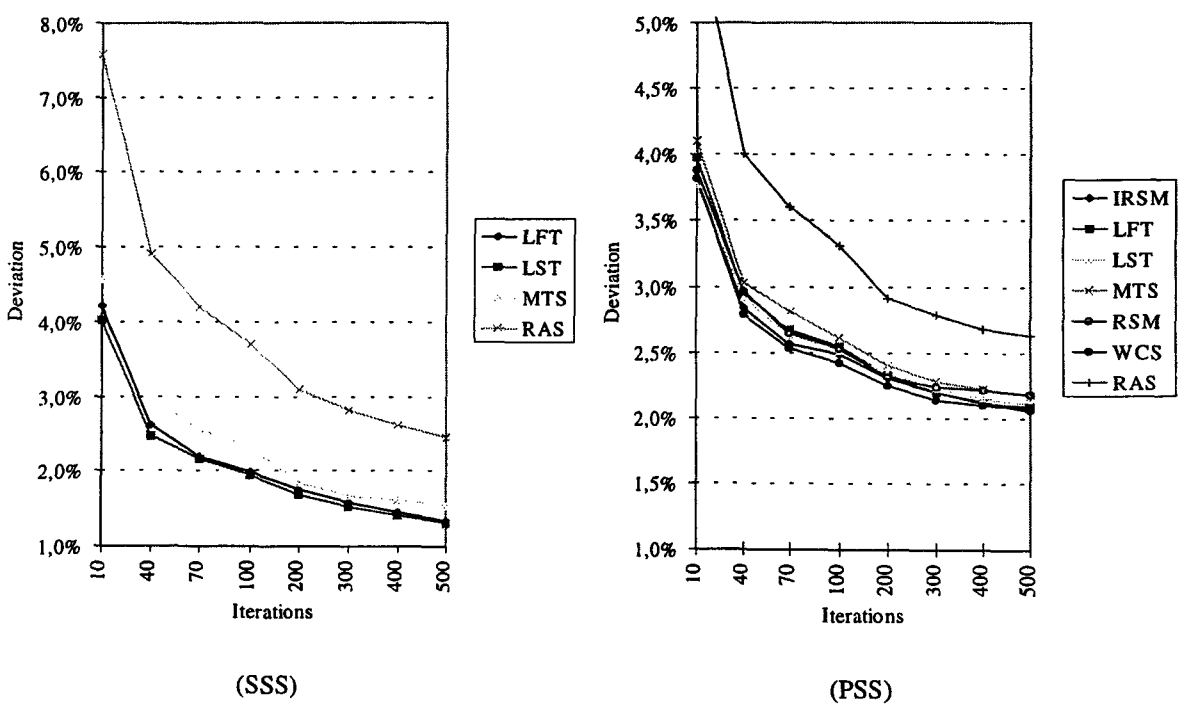


Figure 7: Effect of Iterations and Priority Rules - Deviations

Increasing the number of iterations improves the effectiveness of any of the algorithms used. The deviation for the SSS, using LST, reduces to 1.31%, for the PSS, using IRSM, to 2.06%. Also the observation that linear increases in iterations, and thus computation time, are rewarded by only sublinear decreases of deviation is hardly a surprise, given the complexity of the RCPSP. It is thus especially noteworthy that within certain ranges this relationship is essentially linear. For the PSS, this range lies between 40 and 200 iterations; for the SSS, the deviation reduction remains essentially linear even up to 500 iterations, again due to the fact that the SSS searches the larger solution space than the PSS.

Also, one may note that for the SSS the relative effectiveness of the promising rules is practically immune against changes in the iteration numbers. For the PSS, the picture is somewhat less clear since some rules are more effective than others only within a certain range of iterations; still, from 100 iterations on the differences between the individual rules remain always below 7% relative or 0.14% absolute. We may conclude that good rules remain good, regardless of how many iterations are performed. In particular, the algorithm proclaimed above as best of the ones considered here (SSS, MRBRS/100, LST, $\alpha = 1$) consistently outperforms all other ones, regardless of the number of iterations performed.

5.3. Conclusions

The above findings, limited to the promising rules only, can be summarized as follows.

- *Priority rules:* For the SSS, promising rules are LFT, LST, and MTS, for the PSS IRSM, LFT, LST, MTS, SLK, RSM, and WCS. The bestperforming rules are LST for the SSS and WCS for the PSS. Priority rules performing well when applied deterministically mostly do well also when applied in a sampling manner.

- *Alpha:* For the RBRS $\alpha = 2$ performs best under the SSS, $\alpha = 1$ under the PSS. For the MRBRS/1 $\alpha = 2$, for all other MRBRS-variants $\alpha = 1$ perform best, regardless of the scheduling scheme used.

- *Scheduling schemes:* Using promising rules, the PSS is more effective for small iteration numbers (up to between 40 and 70) while the SSS becomes dominant from there on. Using bad rules, the PSS remains dominant even after 100 iterations.

- *Random sampling schemes:* Under the SSS the MRBRS/10 consistently outperforms all other sampling schemes, except of the MRBRS/100 using LST and $\alpha = 1$ which is the overall best of the tested algorithms. Under the PSS the MRBRS/1 and RBRS are the best schemes; the former should be used if all promising rules, the latter if only the best algorithm is to be utilized.

- *Iterations:* Increasing the number of iterations improves the effectiveness of any of the algorithms used. The deviation of the best serial algorithm reduces to 1.31%, that of the best parallel one to 2.06%. For the PSS, linear increases of the number of iterations between 40 and 200 yield linear improvements of effectiveness; for the SSS, the improvements remain linear even up to 500 iterations. Good algorithms remain good, even if not always best, regardless of how many iterations are performed.

6. Summary

In this contribution, we have studied a number of parameterized biased random sampling schemes, compiling the most influential ones and complementing them by two newly devised schemes. An extensive experimental evaluation, the first to encompass the complete J30 instance set, has been conducted. Restricting the analytical focus to promising priority rules and control parameter values only allowed to explore the effects of the algorithmic components more conclusively than previous studies.

Of the two newly devised sampling schemes, the MRBRS holds significant potential, offering two advantages over existing random sampling schemes. First, in its best configuration the MRBRS/100 improves on the effectiveness of the formerly best sampling method RBRS by 10.20%. Further, practically all serial algorithms using the MRBRS/10 strictly outperform their RBRS-based counterparts - and those based upon any other sampling scheme. Even if our results were derived from using test instances of a project scheduling problem, they have a bearing beyond the realm of project scheduling, since RBRS schemes have been applied to other scheduling problems as well, such as lotsizing and scheduling (Drexl, Haase 1996; Kimms 1996) or staff scheduling (Salewski et al. 1997).

Second, distortions immanent in other sampling schemes, which impinge upon the selection probabilities, are avoided or at least largely reduced. We have shown that e.g. the influence exerted by too large an ε reduces the bias applied by the priorities and thus gears the selection process more towards pure random sampling. Our results demonstrate that, by reducing the adverse effect of ε, the MRBRS schemes allow for better insight into the effects of different priority rules and other factors influencing the performance of sampling heuristics.

Also, our findings bear significance beyond mere priority rule-based methods: since they may form building blocks of more effective - albeit less efficient, due to the problem's complexity - heuristics, we augur that improving them opens new roads for improving more complex heuristics.

Acknowledgements

We are indebted to Andreas Drexl and Rainer Kolisch for their constructive comments on an earlier version of this paper. This work was made possible, in part, by support of the Deutsche Forschungsgemeinschaft (German National Science Foundation) under Grants Dr 170/4-1.

References

ALVAREZ-VALDÉS, R. AND J.M. TAMARIT (1989), "Heuristic algorithms for resource-constrained project scheduling: A review and an empirical analysis", in: Advances in project scheduling, R. Slowinski and J. Weglarz (eds.), Elsevier, Amsterdam, pp. 113-134.

ALVAREZ-VALDÉS, R., J.M. TAMARIT, AND V. VALLS (1989), "Estudio computational de algunos nuevos algoritmos heurísticos para el problema de planificacíon de proyectos con limitacíon de recursos", Trabajos de Investigacíon Operativa 3, No. 1.

BEAN, J.C. (1994), "Genetic algorithms and random keys for sequencing and optimization", ORSA Journal on Computing 6, pp. 154-160.

BEDWORTH, D.D., J.E. BAILEY (1982), *Integrated production control systems - Management, Analysis, Design*, Wiley, New York.

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

BÖTTCHER, J., A. DREXL, R. KOLISCH, AND F. SALEWSKI (1996), "Project scheduling under partially renewable resource constraints", Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel 398.

BOWMAN, E.H. (1959), "The schedule-sequencing problem", Operations Research 7, pp. 621-624.

CARRUTHERS, J.A. AND A. BATTERSBY (1966), "Advances in critical path methods", Operational Research Quarterly 17, pp. 359-380.

CHRISTOFIDES, N., R. ALVAREZ-VALDÉS, R., AND J.M. TAMARIT (1987), "Project scheduling with resource constraints: A branch and bound approach", European Journal of Operational Research 29, pp. 262-273.

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

DAVIS, E.W. (1973), "Project scheduling under resource constraints - Historical review and categorization of procedures", AIIE Transactions (since 1985: IEE Transactions) 5, pp. 297-313.

DAVIS, E.W. AND J.H. PATTERSON (1975), "A comparison of heuristic and optimum solutions in resource-constrained project scheduling", Management Science 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 38, pp. 1803-1818.

DEMEULEMEESTER, E. AND W.S. HERROELEN (1995), "New benchmark results for the resource-constrained project scheduling problem", Research Report 9505, Katholieke Universiteit Leuven, Belgium, to appear in Management Science.

DE WIT, J. AND W.S. HERROELEN (1990), "An evaluation of microcomputer-based software packages for project management", European Journal of Operational Research 49, pp. 102-139.

DREXL, A. (1991), "Scheduling of project networks by job assignment", Management Science 37, pp. 1590-1602.

DREXL, A. AND J. GRÜNEWALD (1993), "Nonpreemptive multi-mode resource-constrained project scheduling", IIE Transactions 25(5), pp. 74-81.

DREXL, A. AND K. HAASE (1996), "Sequential-analysis based randomized-regret-methods for lot-sizing and scheduling", Journal of the Operational Research Society 47, pp. 251-265.

FEO, T.A. AND M.G.C. RESENDE (1989), "A probabilistic heuristic for a computationally difficult set covering problem", Operations Research Letters 8, pp. 67-71.

GAREY, M.R. AND D.S. JOHNSON (1975), "Complexity results for multiprocessor scheduling under resource constraints", SIAM Journal on Computing 4, pp. 397-411.

HAASE, K., J. LATTEIER, AND A. SCHIRMER (1997), "The course scheduling problem at Lufthansa Technical Training", Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel 441.

HART, J.P. AND A.W. SHOGAN (1987), "Semi-greedy heuristics: An empirical study", Operations Research Letters 6, pp. 107-114.

HARTMANN, S. (1997), "A competitive genetic algorithm for resource-constrained scheduling", Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel 451.

HERROELEN, W.S. (1972), "Resource-constrained project scheduling - The state of the art", Operational Research Quarterly 23, pp. 261-275.

JOHNSON, T.J.R. (1967), *An algorithm for the resource-constrained project scheduling problem*, unpublished Ph.D. thesis, Massachusetts Institute of Technology.

KELLEY, J.E. (1961), "Critical path planning and scheduling: Mathematical basis", Operations Research 9, pp. 296-320.

KELLEY, J.E. (1963), "The critical-path method: Resources planning and scheduling", in: *Industrial scheduling*, Muth, J.F. and G.L. Thompson (eds.), Prentice-Hall, Englewood Cliffs, NJ, pp. 347-365.

KHATTAB, M.M. AND F. CHOOBINEH (1991), "A new approach for project scheduling with a limited resource", International Journal of Production Research 30, pp. 185-198.

KIMMS, A. (1996), "Competitive methods for multi-level lot sizing and scheduling: Tabu search and randomized regrets", International Journal of Production Research 34, pp. 2279-2298.

KING, G.W. (1953), "The Monte Carlo method as a natural mode of expression in operations research", Operations Research 1, pp. 46-51.

KOLISCH, R. (1995), *Project scheduling under resource constraints - Efficient heuristics for several problem classes*, Physica, Heidelberg.

KOLISCH, R. (1996a), "Efficient priority rules for the resource-constrained project scheduling problem", Journal of Operations Management 14, pp. 179-192.

KOLISCH, R. (1996b), "Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation", European Journal of Operational Research 90, pp. 320-333.

KOLISCH, R. (1997), "Resource allocation capabilities of commercial project management systems", Research Report, Universität Kiel, Germany.

KOLISCH, R. AND A. DREXL (1996), "Adaptive search for solving hard project scheduling problems", Naval Research Logistics 43, pp. 23-40.

KOLISCH, R. AND A. DREXL (1997), "Local search for nonpreemptive multi-mode resource-constrained project scheduling", to appear in IIE Transactions.

KOLISCH, R. AND R. PADMAN (1997), "An integrated survey of project scheduling - models, algorithms, problems, and applications", Research Report, Universität Kiel, Germany.

KOLISCH, R. AND A. SPRECHER (1997), "PSPLIB - A project scheduling problem library", European Journal of Operational Research 96, pp. 205-216.

KOLISCH, R., A. SPRECHER, AND A. DREXL (1995), "Characterization and generation of a general class of resource-constrained project scheduling problems", Management Science 41, pp. 1693-1703.

KURTULUS, I.S. AND E.W. DAVIS (1982), "Multi-project scheduling: Categorization of heuristic rules performance", Management Science 28, pp. 161-172.

KURTULUS, I.S. AND S.C. NARULA (1985), "Multi-project scheduling: Analysis of project performance", IIE Transactions 17(1), pp. 58-66.

LAGUNA, M., T.A. FEO AND H.C. ELROD (1994), "A greedy randomized adaptive search procedure for the two-partition problem", Operations Research 42, pp. 677-687.

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

LEE, J.-K. AND Y.-D. KIM (1996), "Search heuristics for resource constrained project scheduling", Journal of the Operational Research Society 47, 678-689.

LEON, V.J. AND R. BALAKRISHNAN (1995), "Strength and adaptability of problem-space based neighborhoods for resource constrained scheduling", Operations Research Spektrum 17, pp. 173-182.

MINGOZZI, A., V. MANIEZZO, S. RICCIARDELLI, AND L. BIANCO (1994), "An exact algorithm for project scheduling with resource constraints based on a new mathematical formulation", Research Report 32, Università di Bologna, Italy, to appear in Management Science.

MÜLLER-MERBACH, H. (1967), "Ein Verfahren zur Planung des optimalen Betriebsmitteleinsatzes bei der Terminierung von Großprojekten", Zeitschrift für wirtschaftliche Fertigung 62, pp. 83-88, 135-140 (in German).

NAPHADE, K.S., S.D. WU, AND R.H. STORER (1995), "Problem space search algorithms for the resource-constrained project scheduling problem", Research Report, Lehigh University, Bethlehem, PA.

ÖZDAMAR, L. (1996), "A genetic algorithm approach to a general category project scheduling problem", Research Report, Marmara University, Istanbul, Turkey.

PASCOE, T.L. (1966), "Allocation of resources - CPM", Revue Française de Recherche Opérationelle 38, pp. 31-38.

PATTERSON, J.H. (1973), "Alternate methods of project scheduling with limited resources", Naval Research Logistics 20, pp. 767-784.

PATTERSON, J.H. (1976), "Project scheduling: The effects of problem structure on heuristic procedures", Naval Research Logistics 23, pp. 95-123.

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

PATTERSON, J.H. AND W.D. HUBER (1974), "A horizon-varying, zero-one approach to project scheduling problem", Management Science 20, pp. 990-998.

PATTERSON, J.H. AND G.W. ROTH (1976), "Scheduling a project under multiplie resource constraints: A zero-one programming approach", AIIE Transactions (since 1985: IEE Transactions) 8, pp. 449-455.

PRITSKER, A.A.B., W.D. WATTERS, AND P.M. WOLFE (1969), "Multiproject scheduling with limited resources: A zero-one programming approach", Management Science 16, pp. 93-108.

ROCHAT, Y. AND E.D. TAILLARD (1995), "Probabilistic diversification and intensification in local search for vehicle routing", Journal of Heuristics 1, pp. 147-167.

RUST, J. (1997), "Using randomization to break the curse of dimensionality", Econometrica 65, pp. 487-516.

SALEWSKI, F., A. SCHIRMER, AND A. DREXL (1997), "Project scheduling under resource and mode identity constraints. Model, complexity, methods, and application", to appear in: European Journal of Operational Research.

SCHIRMER, A. (1993), *Audit staff scheduling - Models and methods*, Diploma Thesis, Christian-Albrechts-Universität zu Kiel.

SCHIRMER, A. (1996), "New insights on the complexity of resource-constrained project scheduling - A case of single-mode scheduling", Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel 390.

SPRECHER, A. (1996), "Solving the RCPSP efficiently at modest memory requirements", Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel 425.

SPRECHER, A., R. KOLISCH, AND A. DREXL (1995), "Semi-active, active, and non-delay schedules for the resource-constrained scheduling problem", European Journal of Operational Research 80, pp. 94-102.

STORER, R.H., S.D. WU, AND R. VACCARI (1992), "New search spaces for sequencing problems with application to job shop scheduling", Management Science 38, pp. 1495-1509.

TALBOT, F.B. AND J.H. PATTERSON (1978), "An efficient integer programming algorithm with network cuts for solving resource-constrained scheduling problems", Management Science 24, pp. 1163-1174.

TAVARES, L.V. (1990), "A multi-stage non-deterministic model for project scheduling under resource constraints", European Journal of Operational Research 49, pp. 92-101.

ULUSOY, G. AND L. ÖZDAMAR (1989), "Heuristic performance and network/resource characteristics in resource-constrained project scheduling", Journal of the Operational Research Society 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 Brookes algorithm for project scheduling with resource constraints", Computers and Industrial Engineering 3, pp. 261-268.