Schirmer, Andreas; Riesenberg, Sven

Class-based control schemes for parameterized project scheduling heuristics

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 471

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

Manuskripte
aus den
Instituten für Betriebswirtschaftslehre
der Universität Kiel

No. 471

Class-Based Control Schemes for Parameterized Project Scheduling Heuristics

Schirmer, Riesenberg

Manuskripte
aus den
Instituten für Betriebswirtschaftslehre
der Universität Kiel

No. 471

Class-Based Control Schemes for Parameterized Project Scheduling Heuristics

Schirmer, Riesenberg

April 1998

Andreas Schirmer [a,b,*], Sven Riesenberg [b]

[a] Institut für Betriebswirtschaftslehre, Lehrstuhl für Produktion und Logistik,
Christian-Albrechts-Universität zu Kiel, Wilhelm-Seelig-Platz 1, D-24098 Kiel, Germany
[b] Institut für Informatik und Praktische Mathematik, Lehrstuhl für Systeme zur Informationsverarbeitung,
Christian-Albrechts-Universität zu Kiel, Hermann-Rodewald-Straße 3, 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: Most scheduling problems are notoriously intractable, so the majority of algorithms for them are heuristic in nature. Priority rule-based methods still constitute the most important class of these heuristics. Of these, in turn, parameterized biased random sampling methods have attracted particular interest, due to the fact that they outperform all other priority rule-based methods known. Yet, even the 'best' such algorithms are unable to relate to the particularities of all possible instances of the problem at hand: usually there will exist instances on which other, e.g. the second- or third-best, algorithms perform better. We maintain that asking for the one best algorithm for a given problem may in fact be asking too much. The recently proposed concept of *control schemes*, which refers to algorithmic schemes allowing to guide the proceeding of parameterized algorithms, opens up ways to refine existing algorithms in this regard. By partitioning the set of all instances of a problem into equivalence classes and identifying algorithmic components that are suited for the respective classes, *class-based control schemes* constitute one way to achieve this goal. Using the resource-constrained project scheduling problem as a vehicle, we describe how to devise such control schemes, making systematic use of different scheduling schemes as well as random sampling schemes and priority rules. Results from extensive computational experimentation validate effectiveness and efficiency of our approach.

Keywords: PROJECT SCHEDULING; PARAMETERIZED SAMPLING HEURISTICS; PRIORITY RULES; CONTROL SCHEMES

"[S]cheduling problems often exhibit unique characteristics which make them more amenable to solution with a particular procedure. The advantages of analyzing problem structure and then choosing a technique for solving it can be significant." (Patterson 1976)

1. Introduction

Most combinatorial scheduling problems belong to the class of strongly **NP**-equivalent problems (Garey, Johnson 1979) and thus are notoriously intractable: if one insists on algorithms guaranteed to find an optimal solution, the current state of the art has only exponential answers to offer. It therefore comes as no surprise that the majority of scheduling algorithms devised are heuristic in nature. Of these, priority rule-based methods - despite their age - still constitute the most importantclass. Kolisch (1996b) offers several arguments for their popularity. First, they are easily used and implemented. Actually, most commercial scheduling software relies on simple priority rules (De Wit, Herroelen 1990; Kolisch 1997). Second, they are computationally inexpensive in terms of computer time and memory required, which makes them amenable even for large instances of computationally intractable problems. Third, this very inexpensiveness allows to integrate them as fast "subroutines" into more complex metaheuristic algorithms. 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 (Kolisch 1996b; Schirmer, Riesenberg 1997). Different random sampling schemes have been proposed of which the regret-based scheme of Drexl (1991) and the modified schemes of Schirmer (1997) are the best-performing ones currently known.

Yet, even such algorithms are unable to relate to the particularities of all possible instances of the problem at hand. Even if the proclamation of an algorithm as 'best' is based upon extensive

experimentation, such a recommendation nevertheless reflects the best average performance over all instances attempted. Still, there will usually exist instances on which other, e.g. the second- or third-best, algorithms perform better than the 'on average' best one. We maintain that asking for the one best algorithm for a given problem may in fact be asking too much. Some theoretical foundation of this view, even if pertaining to search algorithms only, has been provided in terms of the so-called no-free-lunch theorem by Wolpert, Macready (1995). The main thrust of the paper is that all search algorithms, if their construction meets certain requirements, perform the same when considering all instances of a problem. The consequence is that if some instances of a problem are better solved by an algorithm A, then some of the remaining instances of the same problem must be solved better by an algorithm B. Although the proof of the theorem makes certain idealizing assumptions, e.g. that points in the space searched are not revisited, which are not easily verified or met in most existing search algorithms, a number of important consequences can be deduced, or at least conjectured, from their theorem (cf. the discussion in Reeves 1997). It stands to reason that analysis of when and why algorithms perform well on some, worse on other instances will receive more and widespread attention.

The concept of *control schemes*, referring to algorithmic schemes which guide the proceeding of parameterized algorithms, opens up ways to take into account such considerations by appropriately refining existing algorithms. By partitioning the set of all instances of a problem into equivalence classes and identifying algorithmic components that are suited for the respective classes, *class-based control schemes* constitute one way to achieve this goal. Using the resource-constrained project scheduling problem (RCPSP) as a vehicle, we describe how to devise such control schemes, making systematic use of different scheduling schemes as well as random sampling schemes and priority rules.

This contribution is in six sections. We briefly discuss the RCPSP in Section 2. The requisite algorithmic components, viz. scheduling schemes, priority rules, random sampling schemes, control schemes, and bounding rules are covered in Section 3. In Section 4 we address the question of how to set up class-based control schemes and graft them onto existing scheduling algorithms. Experimental results of the algorithms arising are presented in Section 5 before Section 6 concludes the paper with some summarizing remarks.

2. Resource-Constrained Project Scheduling

The RCPSP can be characterized as follows: The single project consists of a number J of activities of known duration d_j; all activities have to be executed to complete the project. There are a number R of renewable resources, where the amount available per period is limited by a constant capacity K_r. During each period of its nonpreemptable execution an activity j uses k_{jr} units of resources r. A partial order $\angle$, representing precedence relations between activities,

stipulates that some activities must be finished before others may be started. W.l.o.g. J, R, d_j, K_r (k_{jr}) are assumed to be positive (nonnegative) integers. Let us also assume w.l.o.g. that the activities 1 and J are dummy activities, with durations and resource requirements of zero, and that activity 1 (J) is the unique first (last) activity w.r.t. $\angle$. 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.

The problem 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). Historically, the earliest exact algorithms have used techniques from zero-one programming or dynamic programming. However, in more recent days virtually all approaches were implicit enumeration methods. Also the most powerful exact methods today (Demeulemeester, Herroelen 1997; Mingozzi et al. 1994; Brucker et al. 1996; Sprecher 1996) all belong to this class of algorithms. Yet, owing to the complexity of the RCPSP, most algorithms devised for it 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. Of these, the most effective ones currently available seem to be the tabu search method of Baar et al. (1997), the genetic algorithm of Hartmann (1997), the increasing tolerance algorithm of Hartmann (1998), and the simulated annealing method of Bouleimen, Lecocq (1998). No approximation algorithms are reported in the literature that we are aware of. Detailed reviews of algorithms for the RCPSP, allowing also to trace the historical development, can be found in the survey articles of Herroelen (1972), Davis (1973), Patterson (1984), and Kolisch, Padman (1997).

3. Components of Priority Rule-Based Algorithms

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 according to priorities where more than one activity could be feasibly scheduled. Additional components which have been added successfully to complement these two comprise random sampling schemes, control schemes, and bounding rules. Since deterministic heuristics produce only one - often bad - solution for an

instance, *biased random sampling* methods resolve conflicts according to probabilities proportional to some measure of attractiveness, here the priorities of the candidates. Since proceeding randomly, they may produce several solutions rather than only one as does a deterministic algorithm; the best of these solutions (or one of the best if several were found) is then taken as the result. Evidence gathered in several computational studies confirms that such methods outperform traditional, deterministic approaches (Cooper 1976; Hart, Shogan 1987; Laguna et al. 1994; Drexl, Grünewald 1993). *Control schemes* allow to influence the way in which parameterized algorithms, such as the sampling schemes to be discussed later, are instantiated. Bounding rules, finally, by detecting situations in which a partial or full schedule cannot be improved, restrict the number of schedules to be evaluated and thus improve algorithmic efficiency.

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 (usually also ascribed to Kelley 1963 but actually introduced in an unpublished paper of Brooks in the same year, cf. the remarks in Bedworth, Bailey 1982; Kolisch 1995, p. 68) 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.

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*. Else, 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

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

On each stage, 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 in turn may render some ineligible activities eligible if now all their predecessors are scheduled. The scheme terminates on stage N when all activities are scheduled. The SSS has been studied, among others, by Pascoe (1966), Müller-Merbach (1967), Cooper (1976), Boctor (1990), Valls et al. (1992), and Kolisch (1996b). Note that the SSS searches among the set of active schedules which always contains at least one optimal schedule for any RCPSP-instance considered.

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 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. 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), Ulusoy, Özdamar (1989), Boctor (1990), 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). Formal descriptions of both schemes can be found in Schirmer (1997).

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 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 in the formal descriptions in the Appendix) can be expressed as

$$j^* \leftarrow \min \{ j \in D_n \mid v(j) = \text{extr} \{ v(j') \mid j' \in D_n \} \} \qquad (2)$$

We briefly introduce several well-known priority rules. Let for each activity j ($1 \leq j \leq J$) denote S_j the set of all *immediate* successors w.r.t. $\prec$ and $EFST_j$ the - dynamically updated -

earliest feasible start time w.r.t. *all* constraints. Using this notation, the rules can be defined as done in Table 1 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). These rules were selected because they have been found to be the best-performing ones in several studies (Davis, Patterson 1975; Alvarez-Valdés, Tamarit 1989; Ulusoy, Özdamar 1989; Boctor 1990; Kolisch 1996a; Schirmer 1997).

The measure $WRUP_j$ has been 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. Concerning the measures $IRSM_j$ and WCS_j, AP_n denotes the set of all pairs of non-identical activities i and j in the decision set whereas E_{ij} denotes the earliest time to schedule activity j if activity i is started at t_n (for details cf. Kolisch 1996a).

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

Table 1: Priority Rules - Definition and Classification

3.3. Biased 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. In the framework considered here, these measures are determined by priority rules. A *biased 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

$$j^* \leftarrow \min \{ j \in D_n \mid j = \pi(k) \wedge \zeta \leq \sum_{i=1}^{k} p(\pi(i)) \} \tag{3}$$

One of the two schemes that we use here is the *regret-based biased random sampling* (RBRS) one presented in Drexl, Grünewald (1993). 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

$$v'(j) \leftarrow \begin{cases} \max V(D_n) - v(j) & \text{iff extr = min} \\ v(j) - \min V(D_n) & \text{iff extr = max} \end{cases} \qquad (j \in D_n) \tag{4}$$

and modified by

$$v''(j) \leftarrow (v'(j) + \varepsilon)^{\alpha} \qquad (j \in D_n) \tag{5}$$

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

$$p(j) \leftarrow \frac{v''(j)}{\sum_{j' \in D_n} v''(j')} \qquad (j \in D_n) \tag{6}$$

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 $\alpha>1$, respectively. 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.

The second scheme (MRBRS) we use is a modification of the RBRS, in that it computes regrets from the original priorities according to (4) and modifies them according to (5). How-

ever, rather than using a constant value of ε, ε is determined dynamically. Letting denote $V'(D_n)^+$ the set of all positive transformed priorities of the candidates in D_n, i.e.

$$V'(D_n)^+ \leftarrow \{v'(j) \mid j \in D_n \wedge v'(j) > 0\} \tag{7}$$

ε can be derived from

$$\varepsilon \leftarrow \begin{cases} \min V'(D_n)^+ / \delta & \text{iff } (\exists j \in D_n)\ v'(j) > 0 \\ 1 & \text{otherwise} \end{cases} \tag{8}$$

where δ is a positive integer. Selection probabilities are again derived from (6). Thus, the above actually defines a family MRBRS/δ of sampling schemes. This scheme has been found to be particularly effective in conjunction with the SSS. For details cf. Schirmer (1997); Schirmer, Riesenberg (1997).

3.4. Control Schemes

Parameterized algorithms possess (one or more) *control parameters* which allow to direct the way in which they proceed. While these parameters are usually understood to be only numerical in nature, we adopt a broader view and include also the choice of certain algorithmic components in the concept. Thus, control parameters in the scheduling algorithms examined here are the scheduling scheme (SS), the priority rule, and the random sampling scheme (RSS) employed, as well as the particular value of α and the iteration number Z used. Note that each combination of scheduling scheme, rule, sampling scheme, and α-value defines one particular scheduling algorithm. To algorithmic schemes which govern the instantiation of the control parameters we refer as *control schemes*. Although this very name has, to the best of our knowledge, not been used before, we maintain that some kind of control scheme is always present whenever parameterized methods are used. We outline what we see as the most simple - and most common - form of control scheme before we describe how this simple approach can be refined.

Fixed control schemes (FCS) constitute the most simple class of control schemes. The attribute 'fixed' indicates that the way in which the control parameters are instantiated is prescribed in advance and for all instances alike. As an example, consider a numerical control parameter $\alpha \in \mathbb{N}$ of an algorithm that is to be run for three iterations. Under a FCS, α could e.g. be instantiated by the same value thrice or by three different values, one in each iteration. Based upon the results of previous experimentation, these precepts will reflect those control parameter instantiations which produced the best results on average over all test instances used. FCS are e.g. used by Kolisch (1996b) and Böttcher et al. (1996).

Another approach is motivated by the observation that, while being appealing to try and find 'the best algorithm' for a problem, for most computationally intractable problems there simply exists no algorithm which performs best on all instances. Thus, as the prescriptions of FCS apply to all instances alike, they are unable to take into account the specifics of particular instances. This observation has already been made, among others, by Davis, Patterson (1975). Therefore, a more refined class of schemes, to which we refer as *class-based control schemes* (CCS), proceeds by instantiating the control parameters depending on some characteristics of the instances attempted. Such schemes utilize a partition of the problems instances into equivalence classes and prescribe the application of specific parameter instantiations for each of the classes. Kolisch, Drexl (1996) propose a control scheme which selects the scheduling scheme to be applied according to two quantities: the number of iterations to be performed and the resource strength of the instance. Another example of a CCS is found in Kimms (1997). Also these schemes are based upon extensive experimentation to develop sufficient insight into the influence of the parameters on the algorithms' performance.

The advantage of refining an algorithm from a fixed to a class-based control scheme is obvious since doing so cannot deteriorate algorithmic effectiveness but will improve it in most cases. Additional attractiveness lies in the fact that this refinement may be easily accomplished by appropriately exploiting the results used to derive a FCS in the first place. One caveat is in order, though. If the test bed used is relatively small, devising a CCS may result in fine-tuning the algorithm to the specifics of only a few instances; Reeves (1997) graphically refers to such algorithms as "horses for courses". For the sake of robustness, therefore, the test instances upon which the experimentation relies should be as representative of the expressive power of the corresponding problem as possible.

3.5. Bounding Rules

In order to improve the efficiency of heuristic scheduling algorithms, bounds can be employed in two different ways, previously referred to as global or local bounds (Kolisch, Drexl 1996). Local bounds allow to jettison a particular schedule currently under construction; their implementation depends on the scheduling scheme into which they are fit, so we term them *serial* or *parallel* as appropriate. Global bounds can only be applied when a full schedule has been found; they formulate conditions under which a heuristic schedule is optimal and thus allow to terminate the complete run for the instance at hand. Since their implementation is independent of the respective scheduling scheme, we refer to them as *general* bounds.

We extend the above algorithms by incorporating several classical bounding rules (Stinson et al. 1978; Kolisch 1995, pp. 120-127; Kolisch, Drexl 1996), viz. a precedence-based optimality bound, two resource-based, and two time window ones, which latter may be seen as trans-

forming by means of tightening time windows the issue of improving upon a known solution into one of achieving feasibility of the solutions constructed.

General precedence-based lower bound (GPLB) This bound is based upon relaxing the resource constraints such that a lower bound on the project makespan is determined as the length of a critical path in the project network. Let denote FT_j the makespan of the schedule just found. Whenever

$$FT_J = EFT_J \tag{9}$$

the whole run can be terminated. In this case, the heuristically derived upper bound FT_J on the makespan equals the precedence-based lower bound EFT_J such that the schedule is optimal. ■

General resource-based lower bound (GRLB) This bound is based upon relaxing the precedence constraints. Whenever

$$FT_J = \max\left\{ \left\lceil \sum_{j=1}^{J} k_{jr} \cdot d_j) / K_r \right\rceil \mid 1 \le r \le R \right\} \tag{10}$$

holds, the entire run can be terminated. In this case the schedule is optimal because at least one resource is loaded to full capacity, thus further squeezing the makespan is impossible. ■

Serial time window bound (STWB) The fundamental idea of this bound is to decrease the latest finish times of all unscheduled activities whenever an iteration finds an improved upper bound on the makespan. Let denote $\underline{FT}_J$ the makespan of the incumbent best solution and LST_j the latest start time of activity j. Whenever a feasible solution with a shorter makespan of $FT_J < \underline{FT}_J$ has been found, updated LST-values can be determined for all activities by

$$LST'_j \leftarrow LST_j - \underline{FT}_J + FT_J - 1 \qquad (1 \le j \le J) \tag{11}$$

Now, the current iteration can be terminated whenever a selected activity j* would have to start outside its updated time window, i.e.

$$EFST_{j*} > LST'_{j*} \tag{12}$$

the reason being that the incumbent partial schedule could not be completed to a full schedule with a makespan better than the best known one. By construction of the STWB, any feasible schedule will be better than the previous one since its makespan will at most equal the makespan of that one, less one. ■

Parallel time window bound (PTWB) With $\overline{T}$ the initial upper bound on the makespan, the current iteration can be terminated whenever

$$t_n > \min \{LST_j \mid j \notin S_n\} - \overline{T} + \underline{FT}_J - 1 \tag{13}$$

holds[1] after incrementing t_n. In this case, as for the STWB, the current partial schedule could not improve on the best known schedule. ■

Parallel resource-based lower bound (PRLB) The current iteration can be terminated whenever

$$t_n + \max \left\{ \left\lceil k_{jr} \left(\sum_{j \in A_n} (FT_j - t_n) + \sum_{j \notin S_n} d_j \right) / K_r \right\rceil \mid 1 \le r \le R \right\} \ge \underline{FT}_J \tag{14}$$

holds after incrementing t_n as then, again, the current partial schedule could not improve on the best known schedule. ■

Another optimality bound is the so-called LB2 of Mingozzi et al. (1994), which by relaxing the nonpreemption and certain precedence constraints identifies maximal sets of activities that could be scheduled in parallel. However, the number of such sets may turn out exponential, and only recently has LB2 become computable for practical purposes by means of column generation (Brucker et al. 1996); due to the effort involved we have chosen not to include it here.

4. Class-Based Control Methods

One may say that the design of CCS follows the same road as other methods - but in the opposite direction. Normally, the design of an algorithm is done before any numerical experiments can be conducted. CCS-based algorithms, in contrast, require the designer to conduct numerical experiments before the control scheme can be put together. Although this view is somewhat incomplete, since the experimentation requires the algorithmic components to be in place beforehand, it still serves to clarify the major difference to the usual proceeding.

In this section we outline the design and the results of our initial experimentation, focussing on the performance of the FCS-based algorithms defined by the algorithmic components described above. Drawing on these results, we propose two CCS-based scheduling algorithms.

---

[1] Although the LST-values could also be decreased in a separate step, similar to (16), the above formula is more efficient as it requires J-N subtractionsless.

4.1. 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 Z. 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 Pentium 133 personal computer 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 comprises four renewable resources and 30 non-dummy activities, having a nonpreemptable duration of between one and ten periods and between one and three successors and predecessors each. 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 NC $\in$ {1.5, 1.8, 2.1}, RF $\in$ {0.25, 0.5, 0.75, 1.0}, and RS $\in$ {0.2, 0.5, 0.7, 1.0}. For each instance cluster defined by a combination of these parameters, J30 contains ten instances, for a total of 480 instances.

Of these, those 120 instances where RS = 1.0 are trivially solvable by the MPM-schedule. Of the remaining sample, the exact algorithm of Demeulemeester, Herroelen (1992) found and verified the optima 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; Kolisch 1996a, b; Kolisch, Drexl 1996; Naphade et al. 1997). The remaining 52 instances were only recently solved to optimality by a modified reimplementation of the above exact algorithm (Demeulemeester, Herroelen 1997), which on average improved the original solutions by 3.65 periods, requiring on average 29 seconds and at most three hours per instance on a 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. Thus, ceteris paribus our deviations from optimum are larger than reported elsewhere since our study includes the 52 rather hard instances unsolved at the time of earlier experimentation.

The factors examined here are scheduling scheme, random sampling scheme, priority rule, and control parameter α. The number of iterations is fixed to 100. The outcome of each 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 Pentium 133 PC with 24 MB RAM under MS DOS 6.0. A full factorial design was used, including also a number of additional priority rules and random sampling schemes; in total our analysis is based upon the results of applying 502 different algorithms to the RCPSP (Schirmer, Riesenberg 1997).

4.2. Preparatory Results

In order to expose the influence exerted by the different factors on the effectiveness and to evaluate the different FCS-based algorithms induced, we detail the results accordingly. In Table 2, all such 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
RBRS	1	3.76%	2.07%	2.07%	3.34%	2.34%	4.92%	3.42%
	2	4.70%	2.02%	1.97%	3.71%	2.32%	7.42%	3.45%
	3	5.47%	2.21%	2.08%	3.83%	2.40%	9.26%	3.60%
MRBRS/1	1	3.37%	2.21%	2.18%	3.15%	2.34%	4.47%	3.59%
	2	3.90%	2.03%	2.00%	3.18%	2.31%	5.90%	4.47%
	3	4.67%	2.10%	1.99%	3.38%	2.36%	7.50%	5.38%
MRBRS/10	1	4.24%	1.99%	1.95%	3.21%	2.31%	6.88%	4.45%
	2	5.26%	2.19%	2.13%	3.69%	2.52%	10.66%	6.15%
	3	6.34%	2.38%	2.17%	3.93%	2.84%	12.63%	7.63%
MRBRS/100	1	4,81%	2,17%	1,89%	3,57%	2,52%	9,74%	5,17%
	2	5,87%	2,24%	2,13%	3,82%	2,64%	12,53%	6,73%
	3	6,47%	2,46%	2,22%	4,00%	2,89%	13,48%	7,81%

Table 2: Effect of Priority Rules - Deviations (SSS)

Underlined are the respective row minima, reflecting the best priority rule for each combination of sampling scheme and α-value. It is evident 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 all sampling schemes considered here LST is the bestperforming rule.

RSS	Alpha	GRPW	IRSM	LFT	LST	SLK	MTS	RSM	SPT	WCS	WRUP
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 3: Effect of Priority Rules - Deviations (PSS)

The corresponding results for the PSS are comprised in Table 3 where the strictly dominated MRBRS/100 is omitted. For all variants of the MRBRS as well as the RBRS with smaller α-values, WCS is the best choice.

A comparative view of the best algorithms for each SS and RSS is provided in Table 4, listing 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. Also, the results for pure random sampling (RAS) are shown. Computation times per 100 iterations over all instances range between 0.51 and 0.6 seconds for the algorithms under the SSS and between 0.4 and 0.57 under the PSS.

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%
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%

Table 4: Effect of Random Sampling Schemes - Summary (Best Algorithms)

4.3. Effect of Bounding

In order to evaluate the effect of the individual bounding rules in detail, we solved all instances with several versions of the best serial and the best parallel FCS-based algorithm, viz. applying no bound at all, each of the applicable bounds separately, and all bounds (except of PRLB, cf. below) together. Additional experimentation demonstrated that the results shown in Table 5 are representative for those of other FCS-based algorithms.

SSS, MRBRS/10, LST	w/o Bounds	GPLB	STWB	GRLB	-	all
CPU [s]	0.61	0.59	0.32	0.60	-	0.31
Reduction [%]	-	3%	47%	2%	-	48%
PSS, RBRS, WCS	w/o Bounds	GPLB	PTWB	GRLB	PRLB	all w/o PRLB
CPU [s]	0.52	0.51	0.35	0.51	1.25	0.34
Reduction [%]	-	2%	32%	2%	-142%	34%

Table 5: Effect of Bounding on the Best Serial and Parallel FCS-Based Algorithms

The only other study addressing the performance of bounding rules in conjunction with priority rule-based scheduling algorithms is covered in Kolisch (1995, pp. 120-127) and Kolisch, Drexl (1996); for comparational purposes the corresponding results are evaluated in Table 6.

SSS, RBRS, LST	w/o Bounds	GPLB	STWB	GRLB	-	all
CPU [s]	4.72	3.42	2.26	4.72	-	2.24
Reduction [%]	-	28%	52%	0%	-	53%
PSS, RBRS, WCS	w/o Bounds	GPLB	PTWB	GRLB	PRLB	all
CPU [s]	2.42	1.97	1.36	2.42	3.06	1.29
Reduction [%]	-	19%	44%	0%	-26%	47%

Table 6: Effect of Bounding in the Study of Kolisch (1995, p. 126)

Whereas the experimentation of Kolisch (1995) found no or even adverse effects of resource-based bounds, i.e. no reduction of the computation times for the GRLB and an increase of about 27% for the PRLB, the reader may recall that it included only 308 of the nontrivial 360 instances of the J30 set. We therefore conducted an additional analysis to find out in which clusters the notorious 52 instances are located. Table 7 reveals that they reside exclusively in clusters where resources are scarcely capacitated (low RS) and in great demand (high RF).

	RS		
RF	0.2	0.5	0.7
0.25	0	0	0
0.50	2	0	0
0.75	23	0	0
1.00	22	4	1

Table 7: Distribution of Notorious 52 Instances of J30 Set

Due to this distribution, one is led to conjecture that the resource-based bounds would work better on the notorious instances. In order to check this, we divided the analysis between the 308 instances used in his study and the 52 instances additionally included in our work in Table

9. However, it turns out that excluding them from the instances attempted hardly changes the effect of the bounding rules. To summarize, our results for STWB and PTWB as well as those including all bounds (except of PRLB) are in line with those reported by Kolisch while the GRLB works slightly better. Somewhat confounding is the poor performance that we found for the GPLB when compared to the excellent performance that Kolisch observed; a result we were unable to reproduce.

SSS, MRBRS/10, LST	w/o Bounds	GPLB	STWB	GRLB	-	all
CPU [s]	0.60	0.58	0.29	0.59	-	0.28
Reduction [%]	-	3%	52%	2%	-	54%
PSS, RBRS, WCS	w/o Bounds	GPLB	PTWB	GRLB	PRLB	all w/o PRLB
CPU [s]	0.49	0.48	0.31	0.48	1.24	0.30
Reduction [%]	-	2%	37%	2%	-152%	39%

Table 8: Effect of Bounding on 308 Instances of J30 Set

SSS, MRBRS/10, LST	w/o Bounds	GPLB	STWB	GRLB	-	all
CPU [s]	0.65	0.64	0.53	0.64	-	0.52
Reduction [%]	-	2%	18%	2%	-	21%
PSS, RBRS, WCS	w/o Bounds	GPLB	PTWB	GRLB	PRLB	all w/o PRLB
CPU [s]	0.67	0.65	0.61	0.65	1.32	0.58
Reduction [%]	-	2%	9%	2%	-98%	14%

Table 9: Effect of Bounding on Notorious 52 Instances of J30 Set

4.4. Algorithmic Design

Since RCPSP-instances that are merely precedence-constrained can be optimally solved in linear time, only the capacity limitations of resources induce the strong **NP**-equivalence of the RCPSP (Schirmer 1996). It is easy to provide a visualization of this theoretical result by disaggregating the results of RAS over RF and RS. This is done in Figure 1, the results of which were derived under the serial version of RAS; the picture remains essentially the same under the parallel one. (The height of the surface represents the average deviation from optimum. The differently shaded stripes reflect different levels of the surface, measured in terms of 1% increments.) Figure 1 indeed demonstrates that less capacitated instances are more tractable than instances with rather scarce resources. Considering the limited amount of information taken into account by priority rules, it seems unreasonable to hope for only one particular rule to outperform all its companions on all problem instances. As we aim at minimizing the project makespan, it is thus straightforward that certain algorithms should perform better on the former instances while other ones should do so on the latter ones.

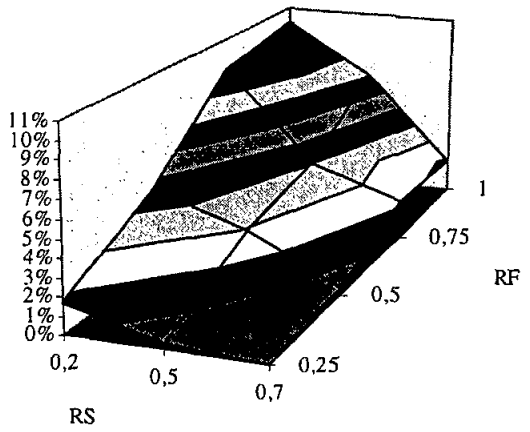
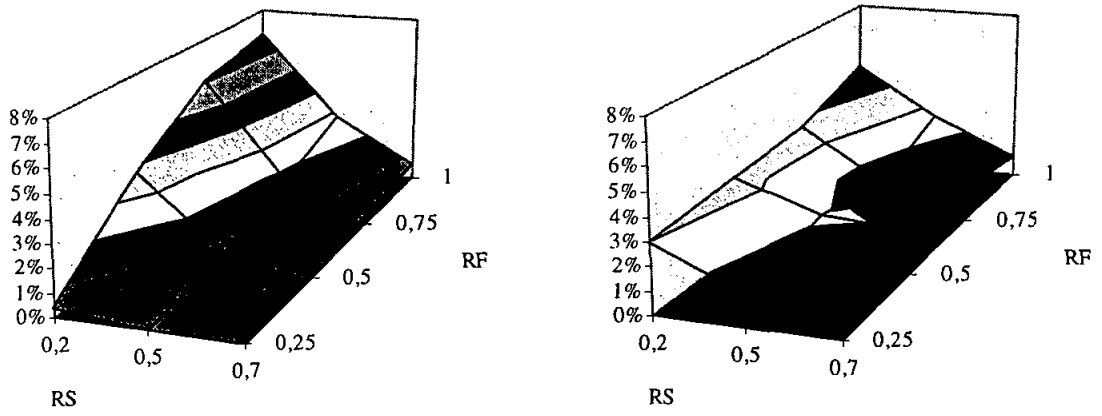


Figure 1: Tractability of Instances as Demonstrated by Effectiveness of RAS

More precisely, we hypothesized that algorithms using precedence-based, and thus objective function-oriented, rules work better on instances where resources are less scarce, i.e. RS is large, and in little demand, i.e. RF is small, and that algorithms using resource-based rules work better on the remaining instances. This hypothesis easily extends from priority rules to other algorithmic components. Kolisch, Drexl (1996) e.g. have demonstrated that the mutual dominance of the serial and the parallel scheduling scheme depends on the resource factor RF of the instance attempted, i.e. that the SSS outperforms its counterpart on instances where RF ≤ 0.75 while the PSS fares better if RF > 0.75.

Accordingly, we analysed all of the FCS-based algorithms separately on all 36 relevant clusters of the J30 instance set; recall that we disregard the trivially solvable 120 instances. As already reported by De Reyck, Herroelen (1996), the network complexity has no significant bearing on algorithmic effectiveness; yet both RF and RS do affect the effectiveness of algorithms. Figure 2 exhibits the respective results of the best serial and parallel FCS-based algorithms, viz. SSS, MRBRS/10, LST and PSS, RBRS, WCS, disaggregated over RF and RS. Note that from now on, we use $\alpha = 1$ which gave the best results for all algorithms discussed here. Clearly, the above hypothesis is supported, as indeed a resource-based algorithm is most effective on instances where resource demand is high compared to resource availability whereas a precedence-based one is most effective on instances where resource scarcity is not a major concern.



(SSS, MRBRS/10, LST) (PSS, RBRS, WCS)

Figure 2: Effect of RF, RS - Deviations

	SSS, MRBRS/10, LST			*PSS, RBRS, WCS*		
	RS			*RS*		
RF	0.2	0.5	0.7	0.2	0.5	0.7
0.25	0.43%	0.00%	0.00%	3.00%	1.36%	1.42%
0.50	3.78%	0.66%	0.08%	3.44%	2.12%	1.91%
0.75	6.07%	1.70%	0.39%	3.70%	1.32%	1.51%
1.00	6.74%	2.81%	0.74%	4.91%	2.67%	0.94%

Table 10: Effect of RF, RS - Deviations

For the sake of devising a CCS-based algorithm, the corresponding numerical results are detailed in Table 10, revealing a clear cut dividing line between those instance clusters on which the best serial FCS-based algorithm dominates and those on which the parallel does (dominated areas are put in shading). Consequently, we propose to combine these two algorithms to a CCS-based scheduling algorithm (CCS-SAR1), as shown in Table 11, by selecting the better of both for each cluster defined by RF and RS.

RF	*RS* 0.2	0.5	0.7
0.25	SSS, MRBRS/10, LST	SSS, MRBRS/10, LST	SSS, MRBRS/10, LST
0.50	PSS, RBRS, WCS	SSS, MRBRS/10, LST	SSS, MRBRS/10, LST
0.75	PSS, RBRS, WCS	PSS, RBRS, WCS	SSS, MRBRS/10, LST
1.00	PSS, RBRS, WCS	PSS, RBRS, WCS	SSS, MRBRS/10, LST

Table 11: Algorithm CCS-SAR1 - Composition

Using the midpoints between the design parameter values to delineate the dividing line between the clusters, a formal description of the algorithm CCS-SAR1 can be given as in Table 12.

Initialization

calculate RF
calculate RS

Execution

if (RF ≤ 0.375
or (0.375 < RF ≤ 0.625 **and** RS ≥ 0.35)
or RS ≥ 0.6)
for z ← 1 **to** Z
call SSS(MRBRS/10, LST, α = 1)
else
for z ← 1 **to** Z
call PSS(RBRS, WCS, α = 1)
endif

Table 12: Algorithm CCS-SAR1

Due to the encouraging effect of bounding on the efficiency of these algorithms, all of the above bounding rules (except of PRLB) were plugged into the algorithm.

5. Experimental Results

5.1. Effectiveness

Figure 3 demonstrates the effectiveness of CCS-SAR1. As is evident from the more flat terrain depicted, using a CCS markedly improves on even the best FCS scheme known for the

problem considered here. The numerical results, provided in Table 13, translate to an average deviation over optimum of 1.53% over all instances attempted. This number compares favorably with the value of 1.89% for the best FCS-based algorithm.

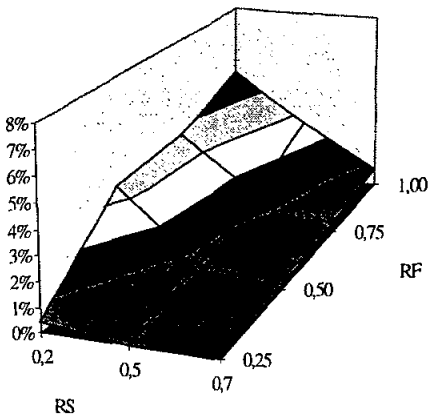


Figure 3: Effectiveness of Algorithm CCS-SAR1

	RS		
RF	0.2	0.5	0.7
0.25	0.43%	0.00%	0.00%
0.50	3.44%	0.66%	0.08%
0.75	3.70%	1.32%	0.39%
1.00	4.91%	2.67%	0.74%

Table 13: Effectiveness of Algorithm CCS-SAR1

5.2. Efficiency

In order to evaluate in more detail the ability of the above bounding rules to improve efficiency, we solved each instance with two versions of the algorithm, employing either none or all of the applicable bounds (except of PRLB). Applicability here refers to the fact that all local, i.e. serial or parallel, bounds were used on certain instance clusters only, as the algorithm applies either a serial or a parallel algorithm to each cluster. Along with the computation times per 100 iterations with bounding rules incorporated, Table 14 shows the effect of bounding in terms of the average reduction of computation times, measured as percentage of the time required without bounding. Also, these numbers indicate the percentage of unnecessary schedules the bounding saves us either from completing in case of a serial or parallel bound or from generating in case of a general bound.

	Efficiency			*Effect of Bounding*		
		RS			*RS*	
RF	0.2	0.5	0.7	0.2	0.5	0.7
0.25	0.30	0.31	0.19	50%	48%	68%
0.50	0.45	0.31	0.19	11%	49%	68%
0.75	0.53	0.28	0.18	12%	49%	70%
1.00	0.59	0.40	0.13	16%	34%	79%

Table 14: Efficiency of and Effect of Bounding Rules on Algorithm CCS-SAR1

5.3. Comparison With Adaptive Search

In order to further demonstrate the effectiveness of our approach, we compare it with the adaptive search procedure (ASP) of Kolisch, Drexl (1996). This procedure is currently the best priority rule-based sampling method available for the RCPSP. Since the authors employed only the 308 instances optimally solved at that time, we exclude the remaining 52 nontrivial instances from consideration to keep the comparison on fair grounds.

Kolisch, Drexl cite results for 100 and 500 iterations. While CCS-SAR1 is based upon the results from running several FCS-based algorithms for 100 iterations, it is shown in Schirmer (1997) that under the PSS the ranking of priority rules changes with changing iteration numbers, albeit under the SSS the ranking remained the same over all iteration numbers. In particular, under the PSS the rule WCS emerges as the best rule for all iteration numbers smaller than 400; for larger samples, the rule LFT fares better, for 400 iterations, both rules perform identically. Also, the dividing line between the "serial" and the "parallel" clusters moves with increasing iteration numbers. Following the same steps as demonstrated above, we therefore devised another control scheme for sample sizes larger than 400, defining an algorithm CCS-SAR2 as shown in Table 15.

	RS		
RF	0.2	0.5	0.7
0.25	SSS, MRBRS/10, LST	SSS, MRBRS/10, LST	SSS, MRBRS/10, LST
0.50	SSS, MRBRS/10, LST	SSS, MRBRS/10, LST	SSS, MRBRS/10, LST
0.75	PSS, RBRS, LFT	SSS, MRBRS/10, LST	SSS, MRBRS/10, LST
1.00	PSS, RBRS, LFT	PSS, RBRS, LFT	SSS, MRBRS/10, LST

Table 15: Algorithm CCS-SAR2 - Composition

The comparison between the algorithms is given in Table 16 where the average, standard deviation, and maximum of the deviations from optimum are listed, along with the percentage of instances solved to optimality. For the sake of completeness, also the corresponding results from considering all 360 nontrivial instances are provided in Table 16.

	100 Iterations			*500 Iterations*		
	ASP	CCS-SAR1		ASP	CCS-SAR2	
Instances	308	308	360	308	308	360
Avg	1.22%	1.05%	1.53%	0.71%	0.67%	1.04%
SD	2.11%	1.85%	2.30%	1.47%	1.44%	1.83%
Max	11.84%	11.11%	11.11%	6.90%	7.89%	10.31%
% Opt	66.23%	67.86%	59.17%	76.62%	65.56%	67.22%

Table 16: Comparison of ASP, CCS-SAR1, and CCS-SAR2

Clearly, the ASP is dominated by suitably devised CCS-based algorithms. Note that the FCS-based algorithms described above are the best FCS-based ones available and that the ASP is the best CCS-based algorithm known so far. Therefore, as our CCS-based approach outperforms both, by transitivity it also dominates the other sampling methods currently available. In order to verify that this relationship extends to higher iteration numbers as well, we have applied several sampling algorithms to all 360 nontrivial instances; the corresponding average deviations are compiled in Table 17.

Algorithm	*Reference*	*Iterations*	
		1000	5000
CCS	-	0.87%	0.59%
Adaptive search	Kolisch, Drexl (1996)	0.99%	0.69%
SSS, MRBRS/10, LST	Schirmer, Riesenberg (1997)	1.09%	0.73%
SSS, MRBRS/10, LFT	Schirmer, Riesenberg (1997)	1.11%	0.75%
PSS, RBRS, WCS	Kolisch (1996a)	1.89%	1.70%
PSS, RBRS, LFT	Kolisch (1996b)	1.86%	1.71%

Table 17: Comparison of Several Sampling Algorithms

6. Summary

In this contribution, we have studied the application of CCS, using the RCPSP as a vehicle. In an extensive computational study, the first to encompass the complete J30 instance set, we have analyzed the best priority rule-based algorithms known for the problem and, drawing upon these results, complemented them by tailored control schemes. The computational re-

sults validate effectiveness and efficiency of our approach. On some instance clusters, the effectiveness of sampling can be improved by up to 2.37% over the best FCS-based algorithm, which amounts to a relative improvement of 39%. Doing so results in a total average deviation, calculated over all instances, of 1.53% after 100 iterations.

Although our findings were derived on test instances of the RCPSP, they do have significance beyond the realm of project scheduling since similar FCS-based algorithms have been applied to other problems such as lotsizing and scheduling (Drexl, Haase 1996; Kimms 1996) or staff scheduling (Salewski et al. 1997). Our results also bear upon other than mere priority rule-based methods since these form building blocks of other (more effective, albeit less efficient, due to the problem's complexity) heuristics: Three of the currently best metaheuristic algorithms for the RCPSP (Hartmann 1997, 1998; Bouleimen, Lecocq 1998) all utilize such methods to generate schedules. We therefore augur that further improving these methods will pave the way also for further improvements in the best known metaheuristics.

Acknowledgements

Parts of this work were conducted while the first author held a teaching position at the Institut für Informatik und Praktische Mathematik, Christian-Albrechts-Universität zu Kiel; we are most grateful for the support, in particular of Peter Kandzia, that made this work possible. We also wish to thank Andreas Drexl for his constructive comments on an earlier version of this paper. Partial funding was granted by the Deutsche Forschungsgemeinschaft (German National Science Foundation) under Grant 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.

BAAR, T., P. BRUCKER, AND S. KNUST (1997), "Tabu search algorithms for the resource-constrained project scheduling problem", Technical Report, University of Osnabrück.

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.

BOULEIMEN, K. AND H. LECOCQ (1998), "A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem", Technical Report, Service de Robotique et Automatisation, Université de Liège, France.

BRUCKER, P., A. SCHOO, AND O. THIELE (1996), "A branch & bound algorithm for the resource-constrained project scheduling problem", Technical Report 178, Osnabrücker Schriften zur Mathematik, University of Osnabrück, to appear in European Journal of Operational Research.

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 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.

DE REYCK, B. AND W.S. HERROELEN (1996), "On the use of the complexity index as a measure of network complexity in activity networks", European Journal of Operational Research 91, pp. 347-366.

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.

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 (1997), "New benchmark results for the resource-constrained project scheduling problem", Management Science 43, pp. 1485-1492.

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.

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

GAREY, M.R. AND D.S. JOHNSON (1979), *Computers and intractability - A guide to the theory of NP-completeness*, W.H. Freeman, San Francisco, CA.

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, to appear in Naval Research Logistics.

HARTMANN, S. (1998), "Acceptance criteria in local search heuristics with an application to project scheduling", Research Report, Universität Kiel, Germany.

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. (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.

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.

KIMMS, A. (1997), "Fallbasiertes Schließen auf Methoden zur Produktionsplanung", Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel 433 (in German).

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 packages", 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 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 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.

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, Universita da 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 (1997), "Problem space search algorithms for resource-constrained project scheduling ", Annals of Operations Research 70, pp. 307-326.

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.

REEVES, C.R. (1997), "Genetic algorithms for the operations researcher", INFORMS Journal on Computing 9, pp. 231-250.

SALEWSKI, F., A. SCHIRMER, AND A. DREXL (1997), "Project scheduling under resource and mode identity constraints: Model, complexity, methods, and application", European Journal of Operational Research 102, pp. 88-110.

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.

SCHIRMER, A. (1997), "Advanced biased random sampling in serial and parallel scheduling", Research Report, Universität Kiel.

SCHIRMER, A., S. RIESENBERG (1997), "Parameterized heuristics for project scheduling - Biased random sampling methods", Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel 456.

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

STINSON, J.P., E.W. DAVIS, AND B.M. KHUMAWALA (1978), "Multiple resource-constrained scheduling using branch and bound", AIIE Transactions 10, pp. 252-259.

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.

WOLPERT, D.H. AND W.G. MACREADY (1995), "No Free Lunch Theorem for Search", Technical Report SFI-TR-95-02-010, Santa Fe Institute, Santa Fe, NM.