Kolisch, Rainer; Drexl, Andreas



# Local search for nonpreemptive multi-mode resource-constrained project scheduling

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 360

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

*Suggested Citation:* Kolisch, Rainer; Drexl, Andreas (1994) : Local search for nonpreemptive multi-mode resource-constrained project scheduling, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 360, Universität Kiel, Institut für Betriebswirtschaftslehre, Kiel

This Version is available at:
https://hdl.handle.net/10419/155429

No. 360

# Local Search for Nonpreemptive Multi-Mode Resource-Constrained Project Scheduling

Kolisch, R. and A. Drexl

December 1994

Rainer Kolisch, Andreas Drexl, Institut für Betriebswirtschaftslehre,
Lehrstuhl für Produktion und Logistik, Christian-Albrechts-Universität zu Kiel,
Olshausenstr. 40, D-24118 Kiel, phone / fax: +49-431-880-1531,
e-mail: kolisch@bwl.uni-kiel.de, drexl@bwl.uni-kiel.de

This paper addresses a general class of nonpreemptive resource-constrained project scheduling problems in which activity durations are discrete functions of committed renewable and nonrenewabe resources. We provide a 0-1 problem formulation and stress the importance of the outlined model by giving applications within production and operations management. Furthermore, we prove that even the problem to derive a feasible solution is ***NP***-complete. As a consequence, solution procedures proposed so far suffer from severe drawbacks: Exact procedures can only solve very small instances to optimality, while heuristic solution approaches fail to generate feasible solutions when problems become highly resource-constrained. Hence, we propose a new local search methodology which first tries to find a feasible solution and second performs a single-neighbourhood search on the set of feasible mode-assignments. In order to evaluate the new procedure we perform a rigorous computational study on the ProGen benchmark-set which is available in the open literature. The experiment includes a comparison of our procedure with other recently proposed heuristics.

---

*Keywords*: Resource-constrained project scheduling; Multiple execution modes; Nonrenewable resource constraints; ***NP***-complete feasibility problem; Local search.

## 1. Introduction

We consider a project consisting of $j$=1,...,$J$ activities, a set of $R$ renewable resources and a set of $N$ nonrenewable resources. Renewable resources have a limited capacity of $K_r \geq 0$ units per period, nonrenewable resources are restricted to $K_r \geq 0$ units for the whole planning horizon. An activity $j$ is executable in one of its $m$=1,...,$M_j$ modes. The mode $m$ determines uniquely the duration $d_{jm}$, the resource usage per period w.r.t. each renewable resource $k_{jmr} \geq 0$, and the resource consumption w.r.t. each nonrenewable resource $k_{jmr} \geq 0$, respectively. Activities are interrelated by technological or so-called precedence relations and by the mutual demand of scarce resources. A feasible solution of the problem is to schedule each activity in one of its modes such that the precedence constraints and all resource constraints are maintained.

The outlined problem is a very general one and embodies a wide range of scheduling problems. Especially, the multi-mode and the single-mode version of the resource-constrained project scheduling problem (RCPSP), job shop and flow shop type problems as well as scheduling problems with one and multiple parallel machines are included. Furthermore, the problem is important within automated manufacturing systems [cf. e.g. Kusiak (1990) and Blazewicz / Finke (1994)]. Here, tasks with alternative process plans are given. Each process plan specifies machines, tools, material handling carriers as well as their sequences. Finally, Kolisch (1994a) shows that the outlined problem also arises within Production Planning and Control systems and Leitstand-systems.

Unfortunately, the problem is intrinsic hard to solve. Incorporating different modes into a given problem setting enlarges the solution space [cf. Ahn / Kusiak (1991) and Li / Willis (1991)] which makes it difficult to find good solutions. Additionally, we will see in the next section that it becomes a formidable task to derive feasible solutions when there are two or more (highly constrained) nonrenewable resources.

On account of this urgent need for solution strategies able to generate feasible and good solutions with a polynomial-bounded effort, we will present a very general local search methodology designed for problems with highly constrained nonrenewable resources. It can

serve as a framework for different local search heuristics and can easily be adapted to handle different objective functions.

The remainder of this paper is organised as follows: In the next Section we will provide a 0-1 programming model of the stated problem and proof the hardness of the feasibility problem for $|N| \geq 2$. Afterwards we will give a review of so-far available solution procedures. In Section 4 we will propose our three-phase solution methodology. Section 5 will report about the results of a detailed computational study where we benchmarked our procedure and three heuristics from the literature on the ProGen instance-set. Section 6 is reserved for final conclusions.

## 2. Model and Complexity Results

In order to model the multi-mode multiple resource-constrained project scheduling problem (MRCPSP) we assume w.l.o.g. that activities are topologically ordered, i.e. each activitiy $j$ has an activity number which is larger than the number of all its immediate predecessors $i \in P_j$ and that we have a unique dummy start and finish activity, $j$=1 and $j$=$J$, each only performable in a single mode associated with zero duration and zero resource demand, respectively. For all other activities we assume that modes are sorted with non-decreasing duration. Given an upper bound $T$ on the processing time of all activities, earliest and latest finish times, $EFT_j$ and $LFT_j$, are calculated for each activity $j$ by traditional forward and backward recursion using the shortest possible durations for activities. Introducing the binary decision variables $x_{jmt} = 1$, if activity $j$ is scheduled in mode $m$ to finish at the end of period $t$, 0 otherwise, we may formulate the following model [cf. e.g. Talbot (1982)]:

$$\sum_{m=1}^{M_j} \sum_{t=EFT_j}^{LFT_j} x_{jmt} = 1 \qquad j=1,...,J \tag{1}$$

$$\sum_{m=1}^{M_i} \sum_{t=EFT_i}^{LFT_i} t\, x_{imt} \leq \sum_{m=1}^{M_j} \sum_{t=EFT_j}^{LFT_j} (t - d_{jm})\, x_{jmt} \qquad j=2,...,J,\ i \in P_j \tag{2}$$

$$\sum_{j=2}^{J-1} \sum_{m=1}^{M_j} k_{jmr} \sum_{\tau=t}^{t+d_{jm}-1} x_{jm\tau} \leq K_r \qquad r \in R,\ t=1,...,T \tag{3}$$

$$\sum_{j=2}^{J-1} \sum_{m=1}^{M_j} k_{jmr} \sum_{t=EFT_j}^{LFT_j} x_{jmt} \leq K_r \qquad r \in N \tag{4}$$

$$x_{jmt} \in \{0,1\} \qquad j=1,...,J,\ m=1,...,M_j,\ t=EFT_j,...,LFT_j \tag{5}$$

Constraint set (1) ensures that each activity $j$=1,...,$J$ is performed in one of its modes and is finished within its time window $[EFT_j, LFT_j]$. Constraints (2) represent the precedence relations. The period capacity of the renewable resource types is maintained by constraint set (3).

The constraints (4) limit the total resource consumption of nonrenewable resources to the available amount. Finally, (5) defines all decision variables to be binary. The commonly used objective function for (1)-(5) is the minimisation of the makespan

$$\text{minimise } \phi = \sum_{t=EFT_J}^{LFT_J} t\, x_{J1t} \tag{6}$$

Other objective functions, e.g. the minimsation of the mean flow time or the maximisation of the net present value can be found in Slowinski (1989).

The problem (1)-(6) is one of the most general and most difficult (project) scheduling problems. As a generalisation of the well-known job shop problem it belongs to the class of ***NP***-hard problems [cf. Blazewicz et al. (1983)]. We will now prove that even the feasibility problem of the MRCPSP is already ***NP***-complete. To do so, we decompose the multi-mode problem into two subproblems: the mode-assignment problem (MAP) and the well-known (single-mode) resource constrained project scheduling problem (RCPSP) [cf. e.g. Patterson / Roth (1976)]. In order to model the MAP, let us introduce the binary decision variables $y_{jm} = 1$ if mode $m$ is assigned to activity $j$, 0 otherwise. We can now state the MAP as follows: Is there a feasible mode-assignment, i.e. a vector of variables $y_{jm}$ such that constraints (7) - (9) hold?

$$\sum_{m=1}^{M_j} y_{jm} = 1 \qquad j=1,...,J \tag{7}$$

$$\sum_{j=1}^{J} \sum_{m=1}^{M_j} k_{jmr}\, y_{jm} \le K_r \qquad r \in N \tag{8}$$

$$y_{jm} \in \{0,1\} \qquad j=1,...,J,\ m=1,...,M_j \tag{9}$$

Constraint set (7) assures that to each activity exactly one of its modes is assigned while the constraints (8) limit the overall consumption of nonrenewable resources to the available amount.

**Theorem**: The MAP is ***NP***-complete for $|N| \ge 2$ and $M_j \ge 2$, $1 < j < J$.

**Proof**: The MAP is in ***NP*** since the feasibility of any mode-assignment $M=(\mu(1),...,\mu(J))$, where mode $\mu(j)$ is assigned to activity $j$, can obviously be checked in polynomial time. To prove that the MAP is ***NP***-complete we polynomially transform the knapsack problem, which is known to be ***NP***-complete [cf. Garey / Johnson (1979)], to the MAP.

The knapsack problem can be stated as follows: Given a finite set $U$ with elements $u$, $1 \le u \le |U|$, a size $s(u) \in \boldsymbol{Z}^+$, a value $v(u) \in \boldsymbol{Z}^+$, for each $u \in U$, a size $B$ and a size $D$, is there a subset $U' \subseteq U$ such that $\sum_{u \in U'} s(u) \le B$ and $\sum_{u \in U'} v(u) \ge D$ hold?

The transformation of the knapsack problem to the MAP is straightforward: Each non-dummy activity represents one element out of $U$ and has two modes, respectively, i.e. $J = |U|+2$,

$M_j$ =2, 1 < $j$ < $J$, and $M_j$ =1, $j \in \{1,J\}$. The subset $U'$ is defined by the activities to which the first mode has been assigned, respectively. Furthermore, the two constraints of knapsack are depicted by two nonrenewable resource constraints, i.e. $|N|$ =2. The first constraint can be employed straightforward. Hence, w.r.t. resource $r$=1, the availability is $K_1$ =$B$, and for each activity $j$=2,...,$J$-1 the resource consumption of the first mode equals $k_{j11}$ =$s(u)$ while the resource consumption of the second mode is zero, i.e. $k_{j21}$ =0. The second knapsack constraint is depicted by multiplying it with "-1". That is, w.r.t. resource $r$=2, the availability arises to $K_2$ = -$D$, and for each activity $j$=2,...,$J$-1 the resource consumption of the first mode equals $k_{j12}$ = -$v(u)$ while the resource consumption of the second mode is zero, i.e. $k_{j22}$ =0. As defined in Section 1, resource availability and consumption are restricted to non-negative values. This requirement is met by adding $|U| \cdot \max\{v(z)|z \in U\}$ to both sides of the second nonrenewable resource constraint. Hence, for resource $r$=2 we end up with an availability of $K_2$ =$|U| \cdot \max\{v(z)|z \in U\}$-$D$. The resource consumption for each activity $j$=2,...,$J$-1 is $k_{j12}$ =$\max\{v(z)|z \in U\}$-$v(u)$ when performed in the first mode and $k_{j22}$ = $\max\{v(z)|z \in U\}$ when done in the second mode. Consequently, each instance of the knapsack problem can be polynomially transformed to an instance of the MAP. A solution of the latter polyniomally converts to a solution of the further as follows: If the first mode has been assigned to activity $j$= 2,...,$J$-1, i.e. $\mu(j)$=1, the corresponding $u$ is in $U'$. This proves the ***NP***-completeness of the MAP and hence the ***NP***-completeness of the feasibility problem of the MRCPSP. ■

## 3. Review of Solution Procedures

### *3.1 Optimal Procedures*

Optimal procedures for solving the MRCPSP have been presented by Talbot (1982), Patterson et al. (1989), Speranza / Vercellis (1993), Sprecher (1994), and Sprecher et al. (1994).

Talbot (1982) was the first who presented an enumeration scheme in order to solve the MRCPSP to optimaltiy. Patterson et al. (1989) refined this method by the introduction of a precedence tree which allowed a systematic enumeration of mode-assignments and start times. Their procedure solved the 536 ProGen benchmark-instances [cf. Subsection 5.1] in an average computation time of 74.10 CPU-seconds on an IBM RS/6000 320 workstation [cf. Kolisch et al. (1995)].

Sprecher (1994) enhanced this approach by four dominance criteria and one feasibility bounding rule. This lowered the computational effort to an average computation time of 0.71 CPU-seconds on the same computer.

Speranza / Vercellis (1993) proposed a depth-first oriented branch and bound procedure with a precedence-based lower bound. It has been shown by Hartmann / Sprecher (1993) that for $|R| \geq 2$ this algorithm may not find the optimal solution and ,moreover, that for instances with $|N| \geq 1$, the algorithm may even be unable to determine an existing feasible solution.

Finally, Sprecher et al. (1994) extended the concept of delay alternatives as employed by Demeulemeester / Herroelen (1992) for the single mode RCPSP to mode- and delay-alternatives. Their implementation solves the ProGen instances in an average of 0.53 CPU-seconds on an IBM-compatible 386-DX personal computer with 40 MHz clockpulse. Despite this en-

couraging results it has to be recalled that exact algorithms in general fail to solve problems with more than 15 activities. Hence, even for small size problems they are of limited use.

*3.2 Heuristic Procedures*

Heuristic solution procedures for the MRCPSP have been provided by Talbot (1982), Drexl / Grünewald (1993), Slowinski et al. (1994), and Özdamar / Ulusoy (1994). Solution methods for the MRCPSP with $|N|=0$, i.e. without nonrenewable resource types, were addressed in Dell'Amico (1990) and Boctor (1994a,b,c).

Talbot (1982) recommends his exact method as truncated enumeration procedure which basically results in a priority-rule-driven serial scheduling method. Drexl / Grünewald (1993) proposed a regret-based biased random sampling approach which jointly employs a serial scheduling scheme and the SPT priority rule. Noteworthy is the fact that the start time of a chosen activity is determined w.r.t. precedence constraints only. Feasibility w.r.t. resource constraints is checked only for the final schedule. The heuristic has been tested extensively on 100 instances with 10 activities, a network complexity of 1.5, three renewable and one nonrenewable resource types as well as different measures of resource scarcity. The best results were obtained with a probability mapping parameter of $\alpha=2$. The sample size - relying on the scarcity of resources - varied between 202 and 3779 in order to generate 100 feasible solutions for each problem, respectively. Depending on the scarcity of resources, an average performance of no more than 3.5% deviation from the optimal objective function value has been documented. The solution time varied between 17 and 321 CPU-seconds on an IBM compatible personal computer with 8086 processor and 10 MHz clockpulse.

Slowinski et al. (1994) proposed a decision support system for a multiple objective MRCPSP which embodies three different solution strategies: A single-pass approach, a multi-pass approach, and simulated annealing. The core of all solution strategies is a precedence-feasible activity list derived with one of 12 priority rules. The single-pass approach deterministically selects the next activity on the list and schedules it in the shortest resource-feasible mode at the earliest period possible. Contrary, the multi-pass approach randomly selects one of the next $c$ precedence-feasible activities of the list for scheduling. Finally, the simulated annealing heuristic employs the activity list in order to represent a solution. The objective function is then calculated by applying the single pass approach. A new neighbour is derived by interchanging the rank of two activities which are not precedence related. The DSS has been mainly proposed as a prototype and hence no rigorous computational experiments are reported.

Very recentyl Özdamar / Ulusoy (1994) presented a constraint-based approach to solve the MRCPSP. They employ a parallel scheduling method in order to decide via so-called essential conditions which activity-mode pairs have to be scheduled. More precisely, at each schedule time, all combinations of feasible activitiy-mode sets are enumerated. A combination consists of precedence-feasible activities performed in exactly one resource-feasible mode. For each combination the increase above a lower bound of the project's makespan is calculated. The combination which induces the smallest increase above the lower bound is then scheduled by applying the essential conditions. The procedure has been tested on a set of 95 instances with

20 to 57 activities, 1 to 6 renewable and one nonrenewable resource type. The constraint-based procedure yielded an average increase over the precedence-based lower bound of 59%. The average computational requirement varied between 20 and 25 CPU-seconds on an IBM 70/386 PS2 computer. Plain priority rules which where applied for comparison purposes yielded an average devitation of 65% within 7-9 CPU-seconds. The constraint-based approach has two disadvantages: First, the worst time complexity of the procedure is exponential. Second, it is not suited for solving MRCPSP's with scarce nonrenewable resource types since it does not provide any special mechanism in order to achieve feasibility w.r.t. constraint type (5).

Boctor (1994a,b,c) developed three solution strategies for the MRCPSP with renewable resource types only. The first heuristic given in Boctor (1994a) is a single-pass approach which employs a parallel scheduling scheme. An activity is in the decision set if all its predecessors are finished and it can be started in at least one of its modes at the current schedule time. Out of the decision set activities are chosen in the order given by the MSLK priority rule and scheduled in the mode with shortest duration. In Boctor (1994b) all possibe activity-mode combinations which can be entirely started at the schedule time are evaluated by applying a lower bound on the increase of the makespan. Finally, Boctor (1994c) performs a simulated annealing procedure similar to the one proposed by Slowinski et al. (1994). On his own set of 240 test problems, with 50 and 100 activities and up to 4 renewable resources, Boctor reports an average percentage deviation from the precedence-based lower bound of 36.8% for the single-pass procedure presented in (1994a), of 34.4% for the heuristic presented in (1994b), and of 26.5% for the simulated annealing method given in (1994c). Computation times are not reported. Nevertheless, while the running time of the single-pass heuristic should be quite modest, the enumeration and evaluation of all possible activity-mode combinations within Boctor (1994b) should take substantial more time. For simulated annealing it is theoretically well-known and has been experimentally shown by van Laarhoven et al. (1992) that convergence to optimality may be achieved at the price of exponential running times.

## 4. A Local Search-Based Heuristic

It has been shown above that the feasibility problem of the MRCPSP belongs to the class of ***NP***-complete problems. Thus, for heuristics it is severely difficult to generate even feasible solutions, a fact which has been pointed out by Drexl / Grünewald (1993). The literature review of the last section revealed that there are currently no heuristics available to handle problems with highly constrained nonrenewable resources. This knowledge has been the primary motivation when designing the solution methodology which will be presented in the sequel. The main idea is to decompose the MRCPSP into two problems: the MAP and the RCPSP. Let the mode-assignment $M$ be a $J$-tuple $M:=(\mu(1),...,\mu(J))$ which assigns to every activity $j$ a unique mode $\mu(j)$. Further, let the schedule $S$ be a $J$-tuple $S:=(FT_1,...,FT_J)$ which assigns to every activity $j$ a unique finish time $FT_j$. Now a solution for the MRCPSP is defined by a mode-assignment $M$ and a schedule $S$ which we will denote by $(M,S)$. We now propose the methodological framework given in Table 1.

**(I) Construction Phase: Generation of an Initial Solution**
Generate an initial mode assignment $M$
Given $M$, generate a schedule $S$ with a fast heuristic for the RCPSP
IF the solution $(M,S)$ is feasible THEN
 Save it as the so-far best solution, $(M^*,S^*):=(M,S)$
ELSE
 GOTO (IV)
**(II) Local Search Phase: Improvement of the Mode-Assignment**
FOR a prescribed number of iterations DO
 Move from the feasible mode-assignment $M$ to a feasible neighbour $M'$
 Given $M'$, generate a schedule $S'$ with a fast heuristic for the RCPSP
 IF $\phi(M',S') < \phi(M^*,S^*)$ THEN
  Save it as the so-far best solution, $(M^*,S^*):=(M',S')$
 SET $M:=M'$
**(III) Intensification Phase: Improvement of the Schedule**
Given $M^*$, generate a schedule $S$ with a near-optimal heuristic for the RCPSP
IF $\phi(M^*,S) < \phi(M^*,S^*)$ THEN
 Save it as the so-far best solution, $(M^*,S^*):=(M^*,S)$
**(IV)** Stop

Table 1: Outline of the Solution Methodology

Following Papadimitriou / Steiglitz (1982) we can outline our solution methodology as follows: In (I), the starting point of the search is generated by the initialisation. A *solution* is represented by a feasible mode-assignment vector $M$. The *cost function* associated with a solution is gained by applying any fast heuristic for the RCPSP to $M$. In (II), a single neighbour of $M$ is generated, evaluated, and stored as current solution. This is done for a prescribed number of neighbour-moves. Finally, in (III) we use the best mode-assignment obtained in (II) in order to derive an improved cost function by means of a near-optimal RCPSP-heuristic.

Before looking at details of our procedure we want to stress the fact that the outlined solution methodology is a very general local search strategy. The way described it is a single neighbourhood search where exactly one neighbour solution is created and evaluated within each pass. Note that a new neighbour replaces the old solution regardless of the objective function. Given this very basic, but also very general framework, we can now apply many local-search-based solution strategies to the MRCPSP.

Let us now focus on two crucial points of the outlined method: The first is the generation of a feasible mode-assignment $M$ within phase (I), the second is the move from a feasible mode-assignment $M$ to a (feasible) neighbour $M'$ within phase (II). Let us start with the first topic.

*4.1 The construction phase: Generation of an initial solution*

An inital mode-assignment $M$ is generated in two basic steps: First, to each of the activities the mode with the smallest relative resource consumption is assigned. Second, the left-over

capacity of nonrenewable resources is allocated to the most promising activities in order to reduce the makespan. Let us use the following notation:

| | |
|---|---|
| $SJ$ | set of activities to which a mode has already been assigned, |
| $NJ$ | set of activities to which a mode has not been assigned, |
| $\pi K_r$ | left-over capacity of the nonrenewable resource $r$, |
| $RC_{jm}$ | relative (nonrenewable) resource consumption of activity $j$ in mode $m$, |
| $SRC_j$ | smallest relative resource consumption of activity $j$, |
| $LRC$ | largest relative resource consumption, |
| $\lambda(j)$ | ordinal position of activity $j$ within priority list $\lambda$ |

The left-over capacity and the relative resource consumption of nonrenewable resources are defined as:

$$\pi K_r = K_r - \sum_{j \in SJ} k_{j\mu(j)r} \tag{10}$$

$$RC_{jm} = \sum_{r \in N} \frac{k_{jmr}}{\pi K_r} \tag{11}$$

We can now give the following formal description of the inital mode-assignment procedure (Init-MA):

Init-MA

**INITIALISATION:** $SJ := \varnothing$; $NJ := \{1,...,J\}$; $\pi K_r := K_r \; \forall \; r \in N$;
**CALCULATE** $RC_{jm}, j \in NJ, m \in \{1,...,M_j\}$;

**(1) FOR** $n$=1 TO $J$ **DO**
$SRC_j := \min\{ RC_{jm} \mid 1 \le m \le M_j \}, j \in NJ$;
$LRC := \max\{ SRC_j \mid j \in NJ \}$;
$j^* := \min\{ j \mid SRC_j := LRC, j \in NJ \}$;
$\mu(j^*) := \min\{ m \mid RC_{j^*m} = SRC_{j^*}, 1 \le m \le M_{j^*} \}$;
$SJ := SJ \cup \{j^*\}$, $NJ := NJ \setminus \{j^*\}$; **UPDATE** $\pi K_r \; \forall \; r \in N, RC_{jm}, j \in NJ, m \in \{1,...,M_j\}$;

**(2) IF** $\pi K_r < 0 \; \exists \; r \in N$, **STOP**
**PERFORM** forward recursion; **SET** $LFT_J := EFT_J$; **PERFORM** backward recursion;
**SET** $s_j := LFT_j - EFT_j$, $1 \le j \le J$; **SORT** activities according to non-decreasing $s_j$;
**STORE** the activity order in the list $\lambda(j)$, $1 \le j \le J$;

**(3) FOR** $n$=1 TO $J$ **DO**
$j := \lambda(n)$; $m := \mu(j)$;
**WHILE** $m \ge 1$ DO
IF $\pi K_r - k_{jmr} + k_{j\mu(j)r} \ge 0 \; \forall \; r \in N$ THEN $\mu(j) := m$;
$m := m-1$;
**UPDATE** $\pi K_r \; \forall \; r \in N$;

**Stop:** **IF** $\pi K_r \ge 0 \; \forall \; r \in N$ **THEN** A feasible mode-assignment $M$ has been generated;

Init-MA works as follows: At the beginning, to none of the activities a mode has been assigned. Consequently the set $SJ$ is empty and the left-over capacity of every nonrenewable re-

source equals its overall capacity, respectively. Afterwards in step (2), for every activity with a not yet assigned mode, the relative resource consumption for each mode is calculated and the activity which maximises the minimum relative resource consumption is selected. To this activity the mode with the smallest relative resource consumption is assigned. In case of ties, the lowest labelled activity and mode are selected, respectively. Finally, the set of activities to which a mode has been assigned and the left-over capacities of nonrenewable resources are updated. The idea why we select the activity which maximises the minimum relative resource consumptions is as follows: From constraint (1) we know that each activity has to be performed in exactly one of its modes. Obviously, the mode with the minimum relative resource consumption is the best way of performing an activity in order to save nonrenewable resource capacity. Hence, by choosing the activity which maximises the minimum relative resource consumption we decrease the left-over capacity of nonrenewable resources by an unavoidable amount as early as possible. This way the scarce resources are reflected properly in order to direct a feasible assignment of the not-yet assigned activities.

After a mode has been assigned to each of the activities, it might be the case that there is still left-over capacity of nonrenewable resources. To allot this capacity efficiently w.r.t. our objective of minimizing the makespan we rank the activities within step (2) in the order of ascending slack as calculated by traditional forward and backward recursion with each activity in the currently assigned mode. Thus, the first activity to be considered has the minimum slack of all activities. Now in step (3), to each activity on the list it is tried to assign the next smaller indexed mode until the first labelled mode is reached or a mode becomes infeasible w.r.t. nonrenewable resource constraints.

**Observation 1**: For $|N|=1$, Init-MA always creates a feasible mode-assignment if one exists.

**Proof**: Obvious, since for each activity the mode with smallest resource consumption is chosen. ■

**Observation 2**: For the nonrenewable resource-unconstrained case, Init-MA assigns to each activity its mode with shortest duration.

**Proof**: Obvious. ■

*4.2 The Local Search Phase: Improvement of the Mode-Assignment*

We define the neighbourhood of a feasible mode-assignment $M$ as follows: For exactly one activity $j$, $1 < j < J$, we change the current mode assigned to $j$ from $\mu(j)$ to $m$, $1 \leq m$, $\mu(j) \leq M_j$, $m \neq \mu(j)$. If the new mode-assignment is feasible we stop with the new neighbour $M'$. Otherwise we carry on changing modes for yet not selected activities. Thereby, to each activity a new mode can only be assigned once. This way we end up with two possible results: Either we obtain a new feasible mode-assignment where to a non propper subset of the activities a new mode has been assigned or we cannot reach a feasible mode-assignment.

The question is now which modes we want to assign to which activities. A quite straightforward way is based on the following considerations: W.r.t. the last schedule generated, for each activity $j$, $1 < j < J$, a modified slack $\sigma_j$ which takes into account precedence as well as resource constraints is calculated. This can be done with the method given by Wiest (1967)

which is reproduced in the Appendix. Now, if the constraints (3) are relaxed, we obtain for a mode-switch of one activity a change of the objective function as follows: Let $[j,m]$ denote the pair of the new mode $m$, $1 \le m \le M_j$, $m \ne \mu(j)$, which can be assigned to activity $j$, $1 < j < J$. Then, the duration difference $\Delta d(j,m)$ induced by assigning mode $m$ to activity $j$ can be written as:

$$\Delta d(j,m) = d_{jm} - d_{j\mu(j)} \qquad 1 \le m, \mu(j) \le M_j, m \ne \mu(j) \tag{12}$$

Under the assumption of relaxed renewable resource constraints the difference for the makespan $\Delta T$ arises to:

$$\Delta T(j,m) = \begin{cases} \Delta d(j,m), \text{ if } \sigma_j = 0 \\ \max\{0, \Delta d(j,m) - \sigma_j\}, \text{ otherwise} \end{cases} \tag{13}$$

Now, the mode-switch associated with the maximum reduction of the makespan is performed.

If the mode-switch causes an infeasible mode-assignment, we have to decrease the resource consumption of at least one other activity. Specifically, activities have to be chosen where a mode-switch frees resources currently causing infeasibility and does not create negative left-over capacity w.r.t. the other nonrenewable resources. Let us define $N'$ to be the subset of nonrenewable resources currently causing infeasibility, i.e. $N' = \{r \mid r \in N, \pi K_r < 0\}$. Then, the number of nonrenewable resource units $\Delta k(j,m)$ freed when assigning mode $m$ to activity $j$ can be written as follows:

$$\Delta k(j,m) = \sum_{r \in N'} (k_{j\mu(j)r} - k_{jmr}) \qquad 1 \le m, \mu(j) \le M_j, m \ne \mu(j) \tag{14}$$

Let us now introduce the following additional notation:

| | |
|---|---|
| $AJ$ | the set of available activities, |
| $AM(j)$ | the available modes of activity $j$, |
| $JM1$ | the activity-mode set 1, i.e. the set of activity-mode pairs $[j,m]$ which can be chosen to improve the objective function value, |
| $JM2$ | the activity-mode set 2, i.e. the set of activity-mode pairs $[j,m]$ which can be chosen to regain feasibility, |

with the following definitions:

$$AM(j) = \{1,\ldots,M_j\} \setminus \mu(j)$$
$$JM1 = \{[j,m] \mid j \in AJ, m \in AM(j)\}$$
$$JM2 = \{[j,m] \mid [j,m] \in JM1, \Delta k(j,m,r) \ge 0 \ \forall r \in N', \Delta k(j,m,r) + \pi K_r \ge 0 \ \forall r \in N \setminus N'\}$$

The procedure Generate New Neighbour (Gen-New-N) can now be presented formally:

Gen-New-N

**Initialisation**: **READ** $M=(\mu(1),...,\mu(J))$; $AJ:=\{1,...,J\}$; **CALCULATE** $AM(j)$, $1 \leq j \leq J$;
**CALCULATE** $\pi K_r \; \forall \; r \in N$; $N':=\varnothing$; **CALCULATE** $JM1$, $JM2$;

(1) **UPDATE** $\Delta T(j,m) \; \forall \; [j,m] \in JM1$;
$\Delta T^{min}:=\min\{\Delta T(j,m) \mid [j,m] \in JM1\}$;
$[j^*,m^*]:=\{[j,m] \mid \Delta T(j,m)=\Delta T^{min}, [j,m] \in JM1\}$;
$\mu(j^*):=m^*$, $AJ:=AJ \setminus \{j^*\}$, **UPDATE** $\pi K_r \; \forall \; r \in N$, **UPDATE** $N'$, $JM1$, $JM2$;
**IF** $N'=\varnothing$ **THEN** STOP;

(2) **IF** $JM2=\varnothing$ **THEN** Stop;
**UPDATE** $\Delta k(j,m) \; \forall \; [j,m] \in JM2$;
$\Delta k^{max}:=\max\{\Delta k(j,m) \mid [j,m] \in JM2\}$;
$[j^*,m^*]:=\{[j,m] \mid \Delta k(j,m)=\Delta k^{max}, [j,m] \in JM2\}$;
$\mu(j^*):=m^*$; $AJ:=AJ \setminus \{j^*\}$; **UPDATE** $\pi K_r \; \forall \; r \in N$; **UPDATE** $N'$, $JM2$;
**IF** $N'=\varnothing$ **THEN** Stop **ELSE GOTO** (2);

**Stop**: **IF** $N'=\varnothing$ **THEN** a new feasible mode-assignment $M$ has been generated;

Gen-New-N starts with every not currently assigned activity-mode pair as an element of the activity-mode set $JM1$. In step (1), the activity-mode pair $[j^*,m^*]$ which - under the assumption of relaxed renewable resource constraints - gives way to the largest improvement of the objective function is chosen out of $JM1$. In case of ties, the lowest activity number is used as the first and the lowest mode number is used as the second tie-breaker. Activity $j^*$ is removed from the set of available activities $AJ$. Afterwards, the left-over capacities of the nonrenewable resources $\pi K_r$, the set of nonrenewable resources currently causing infeasibility $N'$, and the activity-mode sets $JM1$ and $JM2$ are updated, respectively. If the new mode-assignment is feasible, Gen-New-N is finished, otherwise step (2) is performed.

Herein, activity-mode pairs of $JM2$, a subset of $JM1$, are choosable. These are all pairs for which activities have not yet been selected and a mode-change does fulfil the following two nonrenewable resource requirements: It does not increase the resource consumption w.r.t. currently infeasible resources and it does not alter currently feasible resources to become infeasible. The activity-mode pair $[j^*,m^*]$ which frees the most resources currently causing infeasibility is selected. Again, in case of ties the pair with the lowest activity (mode) number is selected. Activity $j^*$ is removed from $AJ$ and updating of the left-over capacities as well as the sets $N'$, $JM1$, and $JM2$ takes place. Step (2) is continued until either feasibility is regained or the set of choosable activity-mode pairs $JM2$ is empty and hence Gen-New-N has to be finished with an infeasible mode-assignment.

*4.3 Numerical Example*

In order to clarify the details of the outlined solution methodology let us consider the example problem provided in Figure 1 and Table 2, respectively.

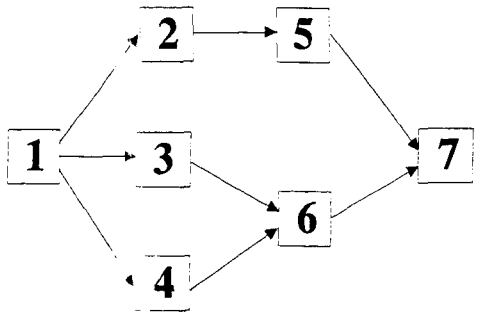


Figure 1: Network of the Example Problem

| $j$ | 1 | 2 | | 3 | 4 | | 5 | 6 | | 7 |
|---|---|---|---|---|---|---|---|---|---|---|
| $m$ | 1 | 1 | 2 | 1 | 1 | 2 | 1 | 1 | 2 | 1 |
| $d_{jm}$ | 0 | 4 | 5 | 2 | 3 | 5 | 2 | 2 | 4 | 0 |
| $k_{jm1}$ | 0 | 2 | 1 | 1 | 3 | 1 | 1 | 2 | 1 | 0 |
| $k_{jm2}$ | 0 | 3 | 1 | 0 | 4 | 2 | 0 | 3 | 2 | 0 |
| $R$={1}, $N$={2}, $K_1$=4, $K_2$=8 | | | | | | | | | | |

Table 2: Data of the Example Problem

| $n$ | $SJ$ | $\pi K_2$ | $RC_{jm}$ ($\underline{SRC_j}$) | | | | | | | | | | | $LRC$ | $j^*$ | $\mu(j^*)$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | $j$ | 1 | 2 | | 3 | 4 | | 5 | 6 | | 7 | | | |
| | | | $m$ | 1 | 1 | 2 | 1 | 1 | 2 | 1 | 1 | 2 | 1 | | | |
| 1 | ∅ | 8 | | 0 | .37 | .12 | 0 | .5 | .25 | 0 | .37 | .25 | 0 | .25 | 4 | 2 |
| 2 | 4 | 6 | | 0 | .5 | .16 | 0 | | | 0 | .5 | .33 | 0 | .33 | 6 | 2 |
| 3 | 4,6 | 4 | | 0 | .75 | .25 | 0 | | | 0 | | | 0 | .25 | 2 | 2 |
| 4 | 2,4,6 | 3 | | 0 | | | 0 | | | 0 | | | 0 | 0 | 1 | 1 |
| 5 | 1,2,4,6 | 3 | | | | | 0 | | | 0 | | | 0 | 0 | 3 | 1 |
| 6 | 1,...,4,6 | 3 | | | | | | | | 0 | | | 0 | | 5 | 1 |
| 7 | 1,...,6 | 3 | | | | | | | | | | | 0 | | 7 | 1 |

Table 3: Deriving the Initial Mode-Assignment within Step (1) of Init-MA

Table 3 reports step (1) of Init-MA. Each row corresponds to one iteration of step (1) where the variable values are given in the columns. Since for the example problem there is $|N|=1$, the mode with smallest resource consumption is assigned to each activity, respectively. Hence, step (1) ends with the mode-assignment $M$=(1,2,1,2,1,2,1).

In step (2), the activities are ranked in the order 1,4,6,7,2,5,3. Table 4 reports the allocation of the remaining 3 nonrenewable capacity units within step (3). Again each row corresponds to one of the $n=J$ iterations. The $\mu(j)$ in the third variable-column represents the initial mode assigned to activity $j$ while the $\mu(j)$ in the sixth variable-column depicts the new mode as-

signed to activity $j$. An empty entry symbolises that the inital mode has not been changed. The final mode-assignment derived by Init-MA is $M$=(1,2,1,1,1,1,1). Applying the single-pass procedure of Kolisch (1994b) in order to solve the remaining RCPSP we obtain the schedule $S$=(0,6,2,5,8,7,8). As can be seen in Figure 2 it has a makespan of 8 periods.

| $n$ | $\pi K_2$ | $j=\lambda(n)$ | $\mu(j)$ | $m$ | $k_{j\mu(j)2}-k_{jm2}$ | $\mu(j):=m$ |
|---|---|---|---|---|---|---|
| 1 | 3 | 1 | 1 | | | |
| 2 | 3 | 4 | 2 | 1 | - 2 | 1 |
| 3 | 1 | 6 | 2 | 1 | - 1 | 1 |
| 4 | 0 | 7 | 1 | | | |
| 5 | 0 | 2 | 2 | 1 | - 2 | |
| 6 | 0 | 5 | 1 | | | |
| 7 | 0 | 3 | 1 | | | |

Table 4: Allocation of the Remaining Capacity Units within Step (3) of Init-MA

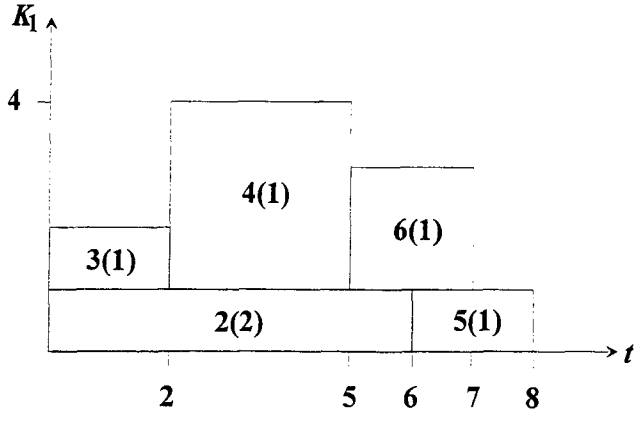


Figure 2: Schedule Associated with the Solution of Init-MA

Starting with the mode-assignment as obtained by Init-MA, the initialisation phase of Gen-New-N terminates with $AJ$={1,...,7}, $\pi K_2$=0, $N'$=∅, $AM(j)$ for all $j$, $1 \le j \le J$, as given in Table 5, $JM1$={[2,1], [4,2], [6,2]}, and $JM2$={[4,2], [6,2]}. Table 6 and Table 7, respectively, report on the mode-assignments obtained by step (1) and step (2) of Gen-New-N. The $\Delta T$-values for the activity-mode pairs in Table 6 were obtained with (13) using the precedence- and resource-based slack values as reported in the Appendix. Now, step (1) of Gen-New-N assigns the 1-st mode to activity 2. Since this leads to infeasibility, step (2) has to be proceeded. Herein, one iteration, switching from the 1-st to the 2-nd mode of activity 4, is sufficient to regain feasibility. Hence, Gen-New-N stops after both step (1) and step (2) have been processed once. The mode-assignment derived is $M$=(1,1,1,2,1,1,1). Solving the remaining RCPSP with the heuristic of Kolisch / Drexl (1994) brings forth the (optimal) schedule $S$=(0,4,2,5,6,7,7) with a makespan of seven periods as depicted in Figure 3.

| $j$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| $AM(j)$ | ∅ | {1} | ∅ | {2} | ∅ | {2} | ∅ |

Table 5: Available Modes after the Initialisation Phase of Gen-New-N

| $\pi K_2$ | $N'$ | $JM1$ | $\Delta T$ | $\Delta T^{min}$ | $[j^*,m^*]$ |
|---|---|---|---|---|---|
| 0 | ∅ | [2,1], [4,2], [6,2] | -2, 1, 1 | -2 | [2,1] |

Table 6: Report of Step (1) of Gen-New-N

| $\pi K_2$ | $N'$ | $JM2$ | $\Delta k$ | $\Delta k^{max}$ | $[j^*,m^*]$ |
|---|---|---|---|---|---|
| -2 | {2} | [4,2], [6,2] | 2, 1 | 2 | [4,2] |

Table 7: Report of Step (2) of Gen-New-N

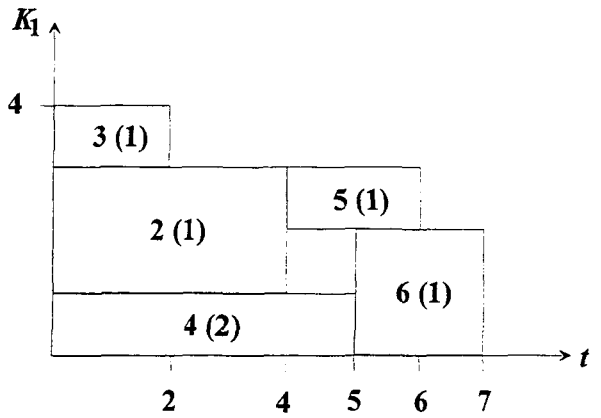


Figure 3: Final Schedule of the Example-Problem

*4.4 Adaption of the Methodology*

The proposed solution methodology can easily be adapted to different objective functions. Furthermore, it may serve as a framework for specific local search-based heuristics.

When seeking for different objective functions, e.g. the maximisation of the net present value, one has to consider the following components of the methodological framework: The fast and the near optimal problem-specicific heuristics employed in phases (I) and (III), respectively, have to be exchanged by procedures for the respective objective function. Furthermore, one may alter the ranking of the activites within Init-MA as well as the presorting of modes. Finally, one has to employ a different criterion than $\Delta T$ in order to choose activity-mode alternatives within step (1) of Gen-New-N.

Examples how specific local seach-based heuristics can be designed with the given methodological framework may be simulated annealing and tabu search. Implementing simulated annealing one may accept a new neighbour only in case of a superior objective function or by an iteration-index-dependent probability, otherwise. In the case of tabu search one may exclude a certain number of recently performed mode-changes from the evaluation.

# 5. Computational Investigation

The computational study which will be presented in this section has been performed with two goals: First, to access the solution quality of the outlined methodology and second, to compare our method with other heuristics for the MRCPSP which have been published in the open literature.

For the computational study we have implemented our solution methodology as a probabilistic local search method by performing step (1) of Gen-New-N with stochastic elements [cf. Goldberg (1989)]. More precisely, we proceeded as follows: Instead of selecting the activity-mode pair which minimised the makespan in step (1) of Gen-New-N, we have used the following mapping function in order to obtain a selection probability for each activity-mode pair out of the set $JM1$ [cf. Kolisch (1994a)]:

$$\psi[j,m] := \frac{(\rho[j,m] + 1)^{\alpha}}{\sum_{[i,n] \in JM1} (\rho[i,n] + 1)^{\alpha}} \tag{15}$$

$\rho[j,m]$ is the maximum makespan-difference $\max\{\Delta T(i,n) | (i,n) \in JM1\}$ subtracted by the makespan-difference $\Delta T(j,m)$ which is realisable when the activity-mode pair $[j,m]$ is chosen, i.e. $\rho[j,m]=\max\{\Delta T(i,n) | (i,n) \in JM1\}-\Delta T(j,m)$. Adding the constant "1" assures that each activity-mode pair $[j,m]$ has a selection probability $\psi[j,m] > 0$. According to preliminary computational results we have set $\alpha=3$. We did not perform the selection of activity-mode pairs within step (2) of Gen-New-N probabilisticly, since preliminary computational results indicated that this strategy does not improve the solution in a significant manner [cf. Kolisch (1994a)].

As RCPSP-heuristics we have chosen a single-pass approach [cf. Kolisch (1994b)] and a hybrid sampling approach with a sample size of 30 schedules [cf. Kolisch / Drexl (1994)]. The choice of this two methods was motivated by the fact that these heuristics seem to be the most competitive suboptimal approaches available today. Nevertheless, it should be noted that any other heuristic can be linked to the local search methodology presented in this paper.

The whole procedure has been coded in PASCAL and implemented on an IBM compatible personal computer with 80386dx processor and 40 MHz clockpulse at the computer laboratory of the Christian-Albrechts-Universität zu Kiel. Random numbers were drawn with the generator proposed by Schrage (1979).

## *5.1 Test Instances*

As test problems we have employed the ProGen instance-set [cf. Kolisch et al. (1995)] which currently serves as the only systematically generated instance-set for scheduling problems with multiple execution modes and different resource categories [cf. Sprecher (1994) and Sprecher et al. (1994)]. The set consists of 640 instances which were generated due to the following three categories of problem parameters: Systematically varied parameters, constant parameters, and randomly out of a specified interval chosen parameters.

Systematically varied parameters are the resource factor *RF* and the resource strength *RS* for renewable and nonrenewable resources, respectively. The resource factor reflects the density of the coefficient matrix given in constraints (3) and (4), respectively. The two levels consid-

ered for each resource factor are 0.5 and 1. The resource strength measures the degree of resource-constrainedness in the interval [0,1]. Generally, $RS$ is computed as follows: $RS=(K_r-K_r^{min})/(K_r^{max}-K_r^{min})$, where $K_r^{min}$ and $K_r^{max}$ are a lower and an upper bound of the resource demand. The four levels considered for each resource strength are 0.2, 0.5, 0.7, and 1.

Constant problem parameters are as follows: $J=12$, $M_j=3$ for $j=2,...,11$ and $M_j=1$ for $j \in \{1,12\}$, $|R|=|N|=2$, $|S_1|=|P_{12}|=3$, and $NC=1.5$, where $S_1$ denotes the set of successor activities of the source activity $j=1$ and $NC$ stands for the network complexity which is the number of nonredundant arcs divided by the number of activities.

Finally, Table 8 reports the parameters which were randomly chosen out of the specified interval. $Q$ symbolises the number of resource types where an activity has a non-zero demand while $U$ denotes the number of units an activity requests from one resource type when having positive demand. More details about the benchmark instances can be found in Kolisch et al. (1995).

| | $d_j$ | $Q_R$ | $U_R$ | $Q_N$ | $U_N$ | $S_j$ | $P_j$ |
|---|---|---|---|---|---|---|---|
| *min* | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| *max* | 10 | 2 | 10 | 2 | 10 | 3 | 3 |

Table 8: Intervals for Randomly Chosen Parameter Levels

Employing a full factorial design study, for each level combination of the systematicly varied problem parameters, 10 instances were randomly generated within the limits given by the other parameter groups. Hence 640 instances emerged where only 536 problems had a feasible solution. Optimal objective function values for each of these instances are reported in Kolisch et al. (1995).

### *5.2 Computational Results*

Table 9 shows the performance of our local search heuristic for different numbers of neighbour-moves (#). Note that each neighbour-move corresponds to one iteration within phase (II).

| # | | 10 | 50 | 100 | 500 |
|---|---|---|---|---|---|
| *DEV* | μ | 10.24 | 5.72 | 4.08 | 1.75 |
| | σ | 13.65 | 9.38 | 7.68 | 4.87 |
| | max | 69.57 | 57.14 | 47.50 | 47.50 |
| *CPU* | μ | 0.06 | 0.17 | 0.29 | 1.22 |
| | σ | 0.04 | 0.05 | 0.07 | 0.31 |
| | max | 0.17 | 0.28 | 0.44 | 1.92 |

Table 9: Effect of the Number of Neighbour-Moves

The computational effort measured in CPU-seconds (*CPU*) increases linearly while the average deviation from the optimal objective function value (*DEV*) improves in an (inverse) exponential manner. That is, good solutions can be obtained very quickly, whereas an extensive computational effort has to be undertaken in order to reach (near) optimal solutions.

In order to compare the performance of our procedure (with #=100) to already published heuristics for the MRCPSP we have coded and implemented the truncated enumeration of Talbot (1982), the biased sampling procedure of Drexl / Grünewald (1993), and the single-pass procedure suggested by Boctor (1994a) under the same conditions as our local search heuristic. For all heuristics the recommendations given by the authors w.r.t. the adjustment of parameters were followed: That is, for the sampling approach due to Drexl / Grünewald (1993) the probability mapping parameter has been set to "2" and a sample size of 100 has been selected. Furthermore, the heuristic has been improved by calculating precedence- and resource-feasible start times for chosen activities [cf. Kolisch (1994a)]. For the single-pass approach of Boctor (1994a), the priority rule MSLK has been implemented. Finally, we imposed a time limit of 10 CPU-seconds on the truncated branch-and-bound procedure of Talbot (1982) when run with the LFT priority rule. Other heuristics are not considered in the study because at the time we undertook the experiments, the methods recently proposed by Boctor (1994b,c) and Özdamar / Ulusoy (1994) were not available. At the end of this section we will give a brief comment on the results to be expected by these procedures.

| | | Boctor (1994a) | | | Talbot (1982) | | | Drexl / Grünewald (1993) | | | Kolisch / Drexl (1994) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | SOL | DEV | CPU | SOL | DEV | CPU | SOL | DEV | CPU | SOL | DEV | CPU |
| $RF_R$ | 0.5 | 50.97 | 12.21 | | 58.96 | 14.88 | | 76.83 | 9.97 | | 100 | 3.76 | |
| | 1 | 52.71 | 17.31 | | 57.04 | 16.89 | | 83.03 | 12.46 | | 100 | 4.37 | |
| $RS_R$ | 0.2 | 58.82 | 23.12 | | 56.30 | 25.18 | | 67.23 | 5.88 | | 100 | 7.98 | |
| | 0.5 | 52.52 | 19.26 | | 60.43 | 19.40 | | 84.17 | 14.40 | | 100 | 4.54 | |
| | 0.7 | 48.55 | 12.54 | | 59.42 | 13.12 | | 83.33 | 14.91 | | 100 | 2.36 | |
| | 1 | 48.57 | 4.04 | | 55.00 | 6.98 | | 83.57 | 7.04 | | 100 | 1.98 | |
| $RF_N$ | 0.5 | 82.33 | 14.85 | | 87.07 | 14.82 | | 97.41 | 8.42 | | 100 | 2.24 | |
| | 1 | 28.26 | 14.97 | | 35.53 | 17.93 | | 66.78 | 13.99 | | 100 | 5.48 | |
| $RS_N$ | 0.2 | 0.00 | - | | 0.00 | - | | 0.00 | - | | 100 | 9.60 | |
| | 0.5 | 24.84 | 22.56 | | 32.68 | 22.74 | | 86.27 | 10.02 | | 100 | 4.91 | |
| | 0.7 | 57.05 | 16.81 | | 69.87 | 22.12 | | 95.51 | 11.03 | | 100 | 3.08 | |
| | 1 | 100 | 11.82 | | 100 | 9.16 | | 98.01 | 6.56 | | 100 | 1.48 | |
| $\mu$ | | 51.86 | 14.89 | 0.00 | 57.83 | 15.91 | 4.40 | 80.00 | 10.88 | 0.45 | 100 | 4.08 | 0.29 |
| $\sigma$ | | | 18.05 | 0.01 | | 18.59 | 4.90 | | 17.07 | 0.03 | | 7.68 | 0.07 |
| max | | | 76.92 | 0.06 | | 86.67 | 10.06 | | 116.6 | 0.55 | | 47.50 | 0.44 |

Table 10: Comparison of Heuristics for the MRCPSP

Additionally to the performance measures *DEV* and *CPU* we denote with *SOL* the percentage of instances for which a feasible solution was derived. Table 10 shows the influence of the

problem parameters on the performance measures *SOL* and *DEV*. With the exception of our procedure for the problem-class $RS_R$, all heuristics show a monotonically deterioration of the performance measure *DEV* for inreasing resource factors and decreasing resource strengths. When the average number of requested nonrenewable resources increases from $RF_N$=0.5 to $RF_N$=1 or when the scarcity of the nonrenewable resources is enforced by lowering $RS_N$, the capability of the other heuristics to derive feasible solutions collapses. None of them succeeds in solving any of the highly nonrenewable resource-constrained instances out of the class $RS_N$=0.2. Even within the problem classes where the other heuristics find feasible solutions for the easier instances, e.g. $RS_N$=0.5 or $RS_N$=0.7, the quality of the solutions in terms of the average deviation from the objective function value is much inferior than the solutions found by our local search procedure on all problems (including the hard ones) of a class. Furthermore, our procedure is faster than the other heuristics except the single-pass approach of Boctor (1994a). This is a benefit of the fact that only feasible mode-assignments are visited and hence no computational time is wasted by generating infeasible solutions.

We can conclude that our local search procedure fulfils three important issues: it derives feasible solutions, the solutions are reasonably close to optimality, and the computational effort is polynomially bounded. Since all other heuristics so-far proposed for the MRCPSP do not explicitly take into account scarce nonrenewable resources, we conjecture that the results of very recently published suboptimal procedures [cf. e.g. Boctor (1994b,c) and Özdamar / Ulusoy (1994)] will show a better performance for the easier problems but will fail again to derive feasible solutions for problems with highly constrained nonrenewable resources. Furthermore, it has to be recalled that the heuristics of Boctor (1994c) and Özdamar / Ulusoy (1994) have worst-case exponential running times.

## 6. Conclusions

A very general project scheduling problem depicting renewable and nonrenewable resource types as well as activities with multiple execution modes has been considered. By transformation of the knapsack problem we showed that already the feasibility problem belongs to the class ***NP***. Hence, we proposed a general local search methodology. It decomposes the overall problem into a mode-assignment problem and a resulting single-mode project scheduling problem. The methodology is characterised by a construction phase which tries to reach an initial solution, a local search phase which performs a single neighbourhood search on the set of feasible mode-assignments, and finally, an intensification phase where on the basis of the best mode-assignment it is tried to find a schedule with an improved objective function. An in-depth computational investigation which included a comparison with other recently proposed heuristics revealed that our local search approach was the only method which arived feasible solution for all problem classes. Futhermore, the performance in terms of the deviation from the optimal objective function value was consistently lowest. On behalf of these encouraging results, refinements of the methodology, e.g. simulated annealing or tabu search applications, should be done in the future in order to derive further improvements in solving general resource-constrained project scheduling problems.

**Acknowledgement**: The authors are indebted to Reimer Karlsen for the coding of the algorithms.

## Appendix: Calculation of Precedence- and Resource-Based Slack

Given a schedule $S$, the procedure slack calculation (Sl-Ca) due to Wiest (1967) calculates a precedence- and resource-based slack $\sigma_j$ for each activity $j$:

Sl-Ca

**Initialisation:** Given a feasible schedule $S=(FT_1,...,FT_J)$,

**SORT** activities according to non-increasing $FT_j$; with 1-st tie-breaker min $d_{j\mu(j)}$,
2-nd tie-breaker min $\sum_{r\in R} k_{jr\mu(j)}$, and final tie-breaker min activity number.

**STORE** the activity order in the list $\lambda(j)$, $1 \le j \le J$;

FOR $n$=1 TO $J$ DO **Stage *n***
BEGIN
$j$:=$\lambda(n)$;
Locally right-shift activity $j$ as far as possible;
Assign to activity $j$ the (new) finish time $FT'_j$;
$\sigma_j$:=$FT'_j$-$FT_j$;
END;

**Stop:** The precedence- and resource-based slack $SL = (\sigma_1,...,\sigma_J)$ has been calculated;

Applying Sl-Ca to the mode-assignment $M$=(1,2,1,1,1,1,1) and the schedule $S$=(0,6,2,5,8,7,8) as obtained for the example problem by phase (I), the results given in Table 11 and 12 are produced.

| $j$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| $FT_j$ | 0 | 6 | 2 | 5 | 8 | 7 | 8 |
| $d_{j\mu(j)}$ | 0 | 6 | 2 | 3 | 2 | 2 | 0 |
| $\lambda(j)$ | 7 | 4 | 6 | 5 | 2 | 3 | 1 |

Table 11: Priority List Obtained by the Initialisation of Sl-Ca

| $n$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| $j=\lambda(n)$ | 7 | 5 | 6 | 2 | 4 | 3 | 1 |
| $FT_j$ | 8 | 8 | 7 | 6 | 5 | 2 | 0 |
| $FT'_j$ | 8 | 8 | 8 | 6 | 6 | 3 | 0 |
| $\sigma_j$ | 0 | 0 | 1 | 0 | 1 | 1 | 0 |

Table 12: Report of Sl-Ca

# References


Ahn, J. and A. Kusiak (1991): Scheduling with alternative process plans, in: Fandel, G. and G. Zäpfel (Eds.): Modern production concepts, Springer, Berlin, pp. 386-403.

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

Blazewicz, J. and G. Finke (1994): Scheduling with resource management in manufacturing systems, European Journal of Operational Research, Vol. 76, pp. 1-14.

Boctor, F.F. (1994a): Heuristics for scheduling projects with resource restrictions and several resource-duration modes, International Journal of Production Research, Vol. 31, pp. 2547-2558.

Boctor, F.F. (1994b): A new and efficient heuristic for scheduling projects with resource restrictions and multiple execution modes, Working Paper No. 94-46, Pavillon des Sciences de l'Administration, Universite Laval, Quebec, Canada.

Boctor (1994c): An adaptation of the simulated annealing algorithm for solving resource-constrained project scheduling problems, Working Paper No. 94-48, Pavillon des Sciences de l'Administration, Universite Laval, Quebec, Canada.

Dell'Amico, M. (1990): Un algoritmo euristico per la pianificazione delle risorse nei progetti software, in: Proceedings of the annual conference of the operational research society of Italy, pp. 211-226.

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

Drexl, A. and J. Grünewald (1993): Nonpreemptive multi-mode resource-constrained project scheduling, IIE Transactions, Vol. 25, No. 5, pp. 74-81.

Garey, M.R. and D.S. Johnson (1979): Computers and intractability - A guide to the theory of NP-completeness, Freeman, San Francisco.

Goldberg, D.E. (1989): Genetic algorithms in search, optimization and machine learning, Addison-Wesley, Reading, Massachusetts.

Hartmann, S. and A. Sprecher (1993): A note on "Hierarchical models for multi-project planning and scheduling", Research Report No. 338, Institut für Betriebswirtschaftslehre, Christian-Albrechts-Universität zu Kiel.

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

Kolisch, R. (1994b): Efficient priority rules for the resource-constrained project scheduling problem, Research Report No. 350, Institut für Betriebswirtschaftslehre, Christian-Albrechts-Universität zu Kiel.

Kolisch, R. and A. Drexl (1994): Hybrid biased random sampling for multiple resource-constrained project scheduling problems, Research Report No. 354, Institut für Betriebswirtschaftslehre, Christian-Albrechts-Universität zu Kiel.

Kolisch, R., A. Sprecher, and A. Drexl (1995): Characterization and generation of a general class of resource-constrained project scheduling problems, to appear in Management Science.

Kusiak, A. (1990): A knowledge- and optimization-based approach to scheduling in automated manufacturing systems, in Brown, D.E. and C.C. White (Eds.): Operations research and artificial intelligence: The integration of problem-solving strategies, Kluwer, Boston, pp. 453-479.

Li, R.K.-Y. and J. Willis (1991): Alternative resources in project scheduling, Computers & Operations Research, Vol. 18, pp. 663-668.

Özdamar, L. and G. Ulusoy (1994): A local constraint based analysis approach to project scheduling under general resource constraints, European Journal of Operational Research, Vol. 79, pp. 287-298.

Papadimitriou, C.H. and K. Steiglitz (1982): Combinatorial optimization - Algorithms and complexity, Prentice-Hall, New Jersey.

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

Patterson, J.H., R. Slowinski, F.B. Talbot, and J. Weglarz (1989): An algorithm for a general class of precedence and resource constrained scheduling problems, in: Slowinski, R. and J. Weglarz. (Eds.): Advances in project scheduling, Elsevier, Amsterdam, pp. 3-28.

Schrage, L. (1979): A more portable Fortran random number generator, ACM Transactions on Mathematical Software, Vol. 5, pp.132-138.

Slowinski, R. (1989): Multiobjective project scheduling under multiple-category resource constraints, in: Slowinski, R. and J. Weglarz (Eds.): Advances in project scheduling, Elsevier, Amsterdam, pp. 113-134.

Slowinski, R., B. Soniewicki, and J. Weglarz (1994): DSS for multiobjective project scheduling subject to multiple-category resource constraints, European Journal of Operational Research, Vol. 79, pp. 220-229.

Speranza, M.G. and C. Vercellis (1993): Hierarchical models for multi-project planning and scheduling, European Journal of Operational Research, Vol. 64, pp. 312-325.

Sprecher, A. (1994): Resource-constrained project scheduling: Exact methods for the multi-mode case, Lecture Notes in Economics and Mathematical Systems, Vol. 409, Springer, Berlin.

Sprecher A., S. Hartmann, and A. Drexl (1994): Project scheduling with discrete time-resource and resource-resource tradeoffs, Research Report No. 357, Institut für Betriebswirtschaftslehre, Christian-Albrechts-Universität zu Kiel.

Talbot, B. (1982): Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case, Management Science, Vol. 28, pp. 1197-1210.

van Laarhoven, P.J.M., E.H.L. Aarts, and J.K. Lenstra (1992): Job shop scheduling by simulated annealing, Operations Research, Vol. 40, pp. 113-125.

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