OR Spektrum (2001) 23: 335–357

OR Spektrum

© Springer-Verlag 2001

Resource–constrained project scheduling: a heuristic for the multi–mode case[*]

Roland Heilmann

Institut für Wirtschaftstheorie und Operations Research, Universität Karlsruhe, Kaiserstraße 12, 76128 Karlsruhe, Germany (e-mail: heilmann@wior.uni-karlsruhe.de)

Received: September 22, 2000 / Accepted: May 18, 2001

Abstract. This paper presents a heuristic solution procedure for a very general resource–constrained project scheduling problem. Here, multiple execution modes are available for the individual activities of the project. In addition, minimum as well as maximum time lags between different activities may be given. The objective is to determine a mode and a start time for each activity such that the temporal and resource constraints are met and the project duration is minimized. Project scheduling problems of this type occur e.g. in process industries. The heuristic is a multi–pass priority–rule method with backplanning which is based on an integration approach and embedded in random sampling. Its performance is evaluated within an experimental performance analysis for problem instances of real–life size with 100 activities and up to 5 modes per activity.

Key words: Resource–constrained project scheduling – Maximum time lags – Multi–mode case – Heuristic

1 Introduction

For projects in practice (e.g. in civil engineering, process industries), it is often possible to perform the individual activities in alternative *modes* which differ w.r.t. processing time, time lags to other activities, and resource requirements. In this way, time–resource and resource–resource tradeoffs (cf. [9]) are reflected. By way of example, we consider two modes of the

[*] This research has been supported by the Deutsche Forschungsgemeinschaft (Grant Ne 137/4).

activity "*installation of a machine*" which belongs to the project "*building a factory*". If this activity is performed in mode 1, it lasts seven hours and two skilled workers are required. If it is executed in mode 2, one skilled and three unskilled workers are needed and the processing time equals four hours.

Because of technological or organizational constraints, *time lags* between the start and completion times of different activities have to be obeyed. In addition to *minimum* time lags, *maximum* ones, which occur e.g. in chemical or food industries, might be given. For example, let us consider the activities "*melting metal*" and "*filling (liquid) metal into mould*" belonging to our example project "*building a factory*". A minimum time lag between these activities has to be observed because the latter can be begun after the completion of the former at the earliest. Moreover, a maximum time lag has to be met: The latter activity has to be completed a certain amount of time after the completion of the former at the latest because otherwise, the metal is no longer liquid enough.

In addition, we have to consider that the capacities of the *resources* which are taken up by the activities are limited. The multi–mode case allows for taking into account limited *renewable* and *nonrenewable* resources. Renewable resources (e.g. machines, manpower) are available at each point in time independently of their utilization formerly whereas nonrenewable resources (e.g. money, energy) are depleted by use.

The planning of these projects can be done w.r.t. alternative objectives such as the maximization of the net present value, the minimization of the resource costs, or the minimization of the duration of the project. Since the latter objective is very important in practice, we focus on it in this paper.

Project scheduling problems of this type can be modelled as instances of the Multi–Mode Resource–Constrained Project Scheduling Problem with minimum and maximum time lags, MRCPSP/max. The objective is to determine a mode and a start time for each activity such that the time lags and the resource constraints are obeyed and the project duration is minimized. Following the classification schemes for project scheduling proposed by [3] and [13], MRCPSP/max is denoted by $MPS \mid temp \mid C_{\max}$ and $m, 1T \mid gpr, mu \mid C_{\max}$, respectively.

MRCPSP/max is in double respect a generalization of resource–constrained project scheduling problems which have been dealt with in literature. On the one hand, MRCPSP/max is more general than the Resource–Constrained Project Scheduling Problem with minimum and maximum time lags, RCPSP/max (cf. [4], [6], [19], [22]). Here, for each activity only one execution mode is available. On the other hand, MRCPSP/max is a generalization of the Multi–Mode Resource–Constrained Project Scheduling Problem, MRCPSP (cf. [14], [15], [24], [25], [26], [27], [28]). Here, only minimum time lags can be taken into account. A survey of existing solution procedures for resource–constrained project scheduling problems can be found in [3].

To the best of our knowledge, apart from two (heuristic) tabu search procedures (cf. [4], [10]) and an (exact) branch–and–bound–procedure (cf. [12]), the literature on MRCPSP/max is completely void. Since these procedures have difficulties w.r.t. the determination of a feasible solution for "large" problem instances (with 100 activities and 3 to 5 modes per activity, cf. Section 5) we devise a new procedure for MRCPSP/max. It is a multi–pass priority–rule method with backplanning.

The remainder of this paper is organized as follows: In Section 2, we state the model for MRCPSP/max. Section 3 is devoted to the solution approach our heuristic is based on. The priority–rule method itself is presented in Section 4. An experimental performance analysis follows in Section 5.

2 Model

Let a project consist of a finite set $V = \{0, 1, \ldots, n, n+1\}$ of activities. Activity 0 represents the start and activity $n+1$ the completion of the project. Both activities are dummy activities.

For each activity $j \in V$ a set $\mathcal{M}_j = \{1, \ldots, |\mathcal{M}_j|\}$ of (execution) modes is available. Activities 0 and $n+1$ can be performed in only one mode: $\mathcal{M}_0 := \mathcal{M}_{n+1} := \{1\}$. Each activity $j \in V$ has to be performed in exactly one mode $\mu \in \mathcal{M}_j$.

The processing time of activity j executed in mode μ is denoted by $p_{j_\mu} \in \mathbb{Z}_{\geq 0}$. The processing time of activities 0 and $n+1$ equals 0: $p_{0_1} := p_{n+1_1} := 0$. S_j and C_j stand for the start time and the completion time (of the performance) of activity j, respectively. If we define $S_0 := 0$, S_{n+1} stands for the project duration. Provided that activity j starts in mode μ at time S_j, it is being executed at each point in time $t \in [S_j, S_j + p_{j_\mu}[$.

Between the start time S_j of activity j, which is performed in mode $\mu \in \mathcal{M}_j$, and the start time S_l of activity l ($l \neq j$), which is performed in mode $\lambda \in \mathcal{M}_l$, a minimum time lag $d^{min}_{j_\mu l_\lambda} \in \mathbb{Z}_{\geq 0}$ or a maximum time lag $d^{max}_{j_\mu l_\lambda} \in \mathbb{Z}_{\geq 0}$ can be given. Note, that a time lag between activity j and activity l depends on mode μ as well as on mode λ. A minimum (maximum) time lag between S_j and C_l, C_j and S_l, and between C_j and C_l can be expressed by a minimum (maximum) time lag between S_j and S_l (cf. [2]).

Activities and time lags are represented by an activity–on–node (AoN) network $N = \langle V, E; \delta \rangle$ with node set V, arc set E, and arc weight function δ. Each element of node set V represents an activity. In the following, we do not distinguish between an activity and the corresponding node. An arc $\langle j, l \rangle \in E$ indicates that a time lag between S_j and S_l has to be observed.

Arc weight function δ assigns to each arc $\langle j,l\rangle \in E$ a $|\mathcal{M}_j| \times |\mathcal{M}_l|$–matrix $(\delta_{j_\mu l_\lambda})_{\mu\in\mathcal{M}_j,\lambda\in\mathcal{M}_l}$ of arc weights as follows: For a minimum time lag $d^{min}_{j_\mu l_\lambda}$ we set $\delta_{j_\mu l_\lambda} := d^{min}_{j_\mu l_\lambda}$. For a maximum time lag $d^{max}_{l_\lambda j_\mu}$ we set $\delta_{j_\mu l_\lambda} := -d^{max}_{l_\lambda j_\mu}$.

$\mathcal{R}^\rho$ denotes the set of renewable resources and $\mathcal{R}^\nu$ the set of nonrenewable resources. $R^\rho_k \in \mathbb{Z}_{>0}$ stands for the capacity of renewable resource k ($k \in \mathcal{R}^\rho$) which is available at each point in time. $R^\nu_k \in \mathbb{Z}_{>0}$ stands for the capacity of nonrenewable resource k ($k \in \mathcal{R}^\nu$) which is available in total. Provided that activity j is performed in mode μ, $r^\rho_{j_\mu k}$ units of renewable resource k ($k \in \mathcal{R}^\rho$) are used at each point in time at which activity j is being executed. Moreover, $r^\nu_{j_\mu k}$ units of nonrenewable resource k ($k \in \mathcal{R}^\nu$) are consumed in total. For activities 0 and $n+1$ we set $r^\rho_{0_1 k} := r^\rho_{n+1_1 k} := 0$ $(k \in \mathcal{R}^\rho)$ and $r^\nu_{0_1 k} := r^\nu_{n+1_1 k} := 0$ $(k \in \mathcal{R}^\nu)$.

Definition 1 *A schedule (M,S) consists of a mode vector M and a start time vector S. A mode vector $M = (m_j)_{j\in V}$ assigns to each activity $j \in V$ exactly one mode $\mu \in \mathcal{M}_j$. A start time vector $S = (S_j)_{j\in V}$ assigns to each activity $j \in V$ exactly one point in time $t \geq 0$ as start time S_j with $S_0 := 0$.*

With $\mathcal{A}(M,S,t) := \{j \in V | S_j \leq t < S_j + p_{j_\mu}\}$ denoting the set of activities being executed at time t for a given schedule (M,S), MRCPSP/max can be stated as follows:

$$\begin{array}{llllll}
\text{Min.} & S_{n+1} & & & & (1)\\
\text{s.t.} & S_l - S_j & \geq & \delta_{j_{m_j} l_{m_l}} & (\langle j,l\rangle \in E) & (2)\\
& \sum\limits_{j\in\mathcal{A}(M,S,t)} r^\rho_{j_{m_j}k} & \leq & R^\rho_k & (k \in \mathcal{R}^\rho;\ t \geq 0) & (3)\\
& \sum\limits_{j\in V} r^\nu_{j_{m_j}k} & \leq & R^\nu_k & (k \in \mathcal{R}^\nu) & (4)\\
& m_j & \in & \mathcal{M}_j & (j \in V) & (5)\\
& S_j & \geq & 0 & (j \in V) & (6)\\
& S_0 & = & 0 & & (7)
\end{array}$$

The objective is to determine a schedule (M,S) such that the time lags (2) are observed, the constraints w.r.t. the renewable resources (3) and the nonrenewable resources (4) are met, and the project duration is minimized (1). Such a schedule is called *optimal*. A schedule (M,S) obeying (2), (3), and (4) is called *feasible*. Since MRCPSP/max is a generalization of RCPSP/max, MRCPSP/max is $\mathcal{NP}$–hard (cf. [2]). Even the corresponding feasibility problem is $\mathcal{NP}$–complete (cf. [12]).

An example for an MRCPSP/max instance is depicted in Figure 1. The project consists of six activities. Activities 1 to 4 can be performed in two modes. (Hence, there exist 16 different mode vectors.) Furthermore, one limited renewable resource and one limited nonrenewable resource are given. Five units of the renewable resource are available at each point in time,

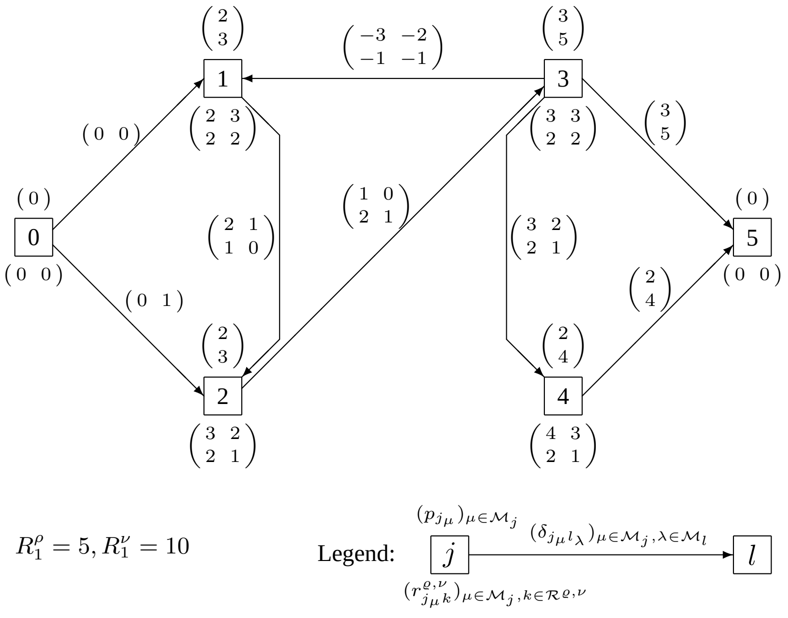
$R_1^\rho = 5, R_1^\nu = 10$
Legend:
$(p_{j_\mu})_{\mu \in \mathcal{M}_j}$
$(\delta_{j_\mu l_\lambda})_{\mu \in \mathcal{M}_j, \lambda \in \mathcal{M}_l}$
$(r^{\varrho,\nu}_{j_\mu k})_{\mu \in \mathcal{M}_j, k \in \mathcal{R}^{\varrho,\nu}}$

Fig. 1. MRCPSP/max instance (P_1)

and ten units of the nonrenewable resource can be consumed in total in the course of the project.

Let us consider the case in which activity 1 is performed in mode 2: The corresponding processing time p_{1_2} equals three units of time. Two units of the renewable resource are used at each point in time at which activity 1 is being executed. Furthermore, two units of the nonrenewable resource are consumed in total. As indicated above, the weights of all incident arcs, i.e. arcs $\langle 1, 2\rangle$, $\langle 0, 1\rangle$, and $\langle 3, 1\rangle$, are mode–dependent. By way of example, let us consider arcs $\langle 1, 2\rangle$ and $\langle 3, 1\rangle$: Provided that activity 2 is performed in mode 1, the performance of activity 2 is allowed to start one unit of time after S_1 at the earliest. If activity 2 is executed in mode 2, its performance must not start before S_1. Provided that activity 3 is executed in mode 1 (2), the performance of activity 3 has to start two units (one unit) of time after S_1 at the latest.

3 Solution approach

In this section, we describe the solution approach of our priority–rule method. It is derived from a division of MRCPSP/max into subproblems.

MRCPSP/max can be divided into two subproblems: Mode assignment problem MAP and RCPSP/max. Subproblem MAP is the problem of determining a mode vector M such that the constraints (4) are observed. Such a mode vector is called *feasible*. MAP is polynomially solvable for $|\mathcal{R}^\nu| = 1$ and $\mathcal{NP}$–complete for $|\mathcal{R}^\nu| \geq 2$ (cf. [14]). Subproblem RCPSP/max is the problem of determining a start time vector S for a feasible mode vector M such that the constraints (2) and (3) are met and the project duration is minimized (1). RCPSP/max is $\mathcal{NP}$–hard, and the corresponding feasibility problem is $\mathcal{NP}$–complete (cf. [2]).

In analogy to MRCPSP (cf. [14], [25]), we distinguish between two solution approaches for MRCPSP/max depending on the sequence in which these subproblems are handled. The *decomposition* approach relies on sequentially dealing with both subproblems: Having solved MAP by determining a feasible mode vector M, we obtain an RCPSP/max instance. Hereafter, we compute a start time vector S by means of a solution procedure for RCPSP/max. The tabu search procedures of [4] and [10] rely on this approach. The *integration* approach is based on simultaneously dealing with MAP and RCPSP/max: The determination of mode vector M and start time vector S is done in parallel. This approach is made use of by the branch–and–bound procedure of [12] and by the priority–rule method presented in this paper.

Similar to the integration approach for MRCPSP (cf. [25]), we *schedule* the activities one after the other. (To schedule an activity means to assign a mode and a start time to it.) Hence, our procedure is a *construction* method. In this way, a sequence of feasible *partial* schedules is created where "partial" indicates that not all activities have been scheduled. For example, w.l.o.g., we assume that activities $0, 1, \ldots, j$ are scheduled one after the other. Then, the respective sequence of partial schedules is $((m_0, 0, \ldots, 0), (S_0, -\infty, \ldots, -\infty))$, $((m_0, m_1, 0, \ldots, 0), (S_0, S_1, -\infty, \ldots, -\infty)), \ldots, ((m_0, m_1, \ldots, m_j, 0, \ldots, 0), (S_0, S_1, \ldots, S_j, -\infty, \ldots, -\infty))$.

But if, different from MRCPSP, maximum time lags have to be observed, it can happen that a given feasible partial schedule cannot be extended to a feasible schedule because at least one nonscheduled activity cannot be scheduled without violating maximum time lags to scheduled activities. In this case, it is necessary to perform a *backplanning* step, i.e., to unschedule some scheduled activities.

In order to detect as early as possible whether a given feasible partial schedule cannot be extended to a feasible schedule (and hence to save computation time), we apply a so–called *feasibility test* (cf. Section 4.3). It makes use of a special relaxation of an MRCPSP/max instance, named *minimal* problem instance (cf. Definition 3), which is based on a *mode set* vector (cf. Definition 2). In addition, this relaxation is needed for the determination of priority–rule values (cf. Section 4.2).

Definition 2 *A mode set vector $\widetilde{\mathcal{M}} = (\widetilde{\mathcal{M}}_j)_{j \in V}$ assigns to each activity $j \in V$ a nonempty set $\widetilde{\mathcal{M}}_j \subseteq \mathcal{M}_j$ of modes.*

The (simultaneous) assignment of several modes to an activity is expressed by a fictitious execution mode. Associated with this special mode are fictitious processing times $\min_{\mu \in \widetilde{\mathcal{M}}_j} p_{j_\mu}$ $(j \in V)$, resource requirements $\min_{\mu \in \widetilde{\mathcal{M}}_j} r^{\varrho,\nu}_{j_\mu k}$ $(j \in V; k \in \mathcal{R}^{\varrho,\nu})$, and arc weights $\delta^{\widetilde{\mathcal{M}}}_{jl} := \min_{\mu \in \widetilde{\mathcal{M}}_j} \min_{\lambda \in \widetilde{\mathcal{M}}_l} \delta_{j_\mu l_\lambda}$ $(\langle j, l \rangle \in E)$. With $\mathcal{A}(\widetilde{\mathcal{M}}, S, t) := \{j \in V | S_j \leq t < S_j + \min_{\mu \in \widetilde{\mathcal{M}}_j} p_{j_\mu}\}$ denoting the set of activities being executed at time t for a given mode set vector $\mathcal{M}$ and start time vector S, we give the following definition.

Definition 3 *Let an MRCPSP/max instance (P) and a mode set vector $\widetilde{\mathcal{M}}$ be given. The corresponding minimal problem instance $(P^{\widetilde{\mathcal{M}}})$ is defined as follows:*

$$\begin{array}{llll}
\text{Min. } S_{n+1} & & & (8)\\
\text{s.t.} & S_l - S_j \geq \min\limits_{\mu \in \widetilde{\mathcal{M}}_j} \min\limits_{\lambda \in \widetilde{\mathcal{M}}_l} \delta_{j_\mu l_\lambda} & (\langle j, l \rangle \in E) & (9)\\
& \sum\limits_{j \in \mathcal{A}(\widetilde{\mathcal{M}}, S, t)} \min\limits_{\mu \in \widetilde{\mathcal{M}}_j} r^{\rho}_{j_\mu k} \leq R^{\rho}_k & (k \in \mathcal{R}^{\rho};\ t \geq 0) & (10)\\
& \sum\limits_{j \in V} \min\limits_{\mu \in \widetilde{\mathcal{M}}_j} r^{\nu}_{j_\mu k} \leq R^{\nu}_k & (k \in \mathcal{R}^{\nu}) & (11)\\
& S_j \geq 0 & (j \in V) & (12)\\
& S_0 = 0 & & (13)
\end{array}$$

$(P^{\widetilde{\mathcal{M}}})$ corresponds to a fictitious (single–mode) RCPSP/max instance where, in addition, constraints (11) w.r.t. nonrenewable resources have to be met. By way of example, the minimal problem instance $(P_1^{\widetilde{\mathcal{M}}})$ corresponding to (P_1) is depicted in Figure 2. $N^{\widetilde{\mathcal{M}}} = \langle V, E; \delta^{\widetilde{\mathcal{M}}} \rangle$ with $\delta^{\widetilde{\mathcal{M}}}_{jl} = \min_{\mu \in \widetilde{\mathcal{M}}_j} \min_{\lambda \in \widetilde{\mathcal{M}}_l} \delta_{j_\mu l_\lambda}$ $(\langle j, l \rangle \in E)$ denotes the network of $(P^{\widetilde{\mathcal{M}}})$. The length of a longest path (*distance*) from activity j to activity l $(j, l \in V)$ in $N^{\widetilde{\mathcal{M}}}$ is denoted by $d^{\widetilde{\mathcal{M}}}_{jl}$. The distance matrix $D^{\widetilde{\mathcal{M}}} = (d^{\widetilde{\mathcal{M}}}_{jl})_{j,l \in V}$ can be computed by means of the Floyd–Warshall algorithm in $\mathcal{O}(|V|^3)$ time (cf. [18], [20]).

Proposition 1 *Let an MRCPSP/max instance (P) and a mode set vector $\widetilde{\mathcal{M}}$ be given. If $\widetilde{\mathcal{M}} = \mathcal{M}$ holds, $(P^{\widetilde{\mathcal{M}}})$ represents a relaxation of (P).*

As indicated above, $(P^{\widetilde{\mathcal{M}}})$ is used in order to find out whether there exists a feasible schedule (M, S) which is based on partial schedule $((m'_0, \ldots, m'_j, 0, \ldots, 0), (S'_0, \ldots, S'_j, -\infty, \ldots, -\infty))$, i.e., where $m_l = m'_l$ and $S_l = S'_l$ $(l = 0, \ldots, j)$. Obviously, to that partial schedule corresponds exactly one

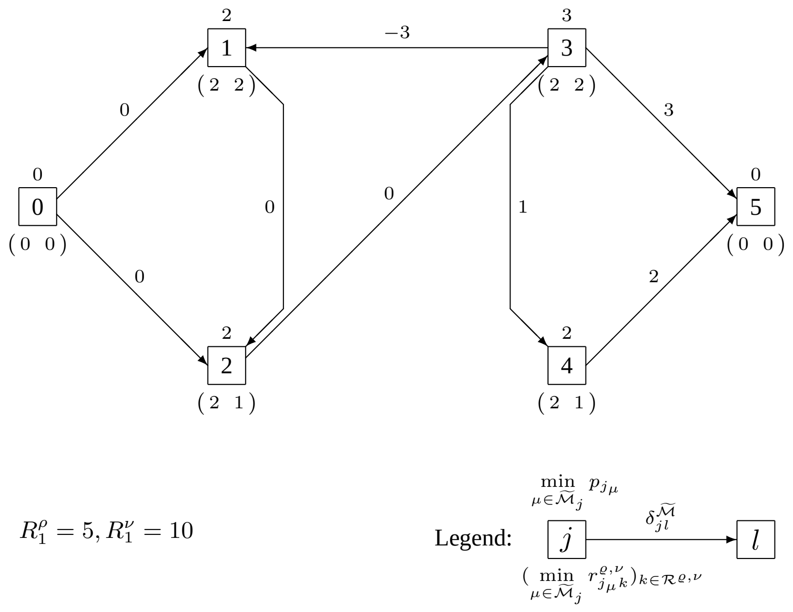
0
1
2
3
4
5
(0 0)
(2 2)
(2 1)
(2 2)
(2 1)
(0 0)
−3
$R_1^\varrho = 5, R_1^\nu = 10$
Legend:
$\min_{\mu \in \widetilde{\mathcal{M}}_j} p_{j\mu}$
$\delta_{jl}^{\widetilde{\mathcal{M}}}$
$(\min_{\mu \in \widetilde{\mathcal{M}}_j} r_{j_\mu k}^{\varrho,\nu})_{k \in \mathcal{R}^{\varrho,\nu}}$
j
l

Fig. 2. Minimal MRCPSP/max instance $(P_1^{\widetilde{\mathcal{M}}})$

mode set vector $\widetilde{\mathcal{M}} = (\{m_0\}, \ldots, \{m_j\}, \{\mathcal{M}_{j+1}\}, \ldots, \{\mathcal{M}_{n+1}\})$. If, using the feasibility test, we can show that the relaxation $(P^{\widetilde{\mathcal{M}}})$ w.r.t. the partial schedule does not have a feasible solution, we know that there does not exist such a feasible schedule (M, S).

4 Priority–rule method

This section is organized as follows: At first, we describe how the priority–rule method works in principle (cf. Section 4.1). Next, we give a detailed description of the essential elements of the procedure: the priority rules (cf. Section 4.2), the feasibility test (cf. Section 4.3), and the backplanning (cf. Section 4.4). Finally, the complete procedure is presented (cf. Section 4.5).

4.1 Basic concepts

As mentioned in Section 3, the procedure is a construction method where the activities are scheduled one after the other. At the beginning, only activity 0 is scheduled. Consequently, the initial mode set vector $\widetilde{\mathcal{M}}$ equals $\mathcal{M}$. At each step of the algorithm, we schedule an activity by assigning a mode and a start time to it. In this way, we try to obtain a feasible schedule (M, S).

When we schedule an activity, we restrict $\widetilde{\mathcal{M}}$ appropriately. Hence, to each step of the procedure belong a current mode set vector $\widetilde{\mathcal{M}}$ and thus a current minimal problem instance $(P^{\widetilde{\mathcal{M}}})$.

The set of scheduled (nonscheduled) activities is denoted by $\mathcal{C}$ ($\overline{\mathcal{C}}$). At the beginning, we set $\mathcal{C} := \{0\}$, $\overline{\mathcal{C}} := V \backslash \{0\}$, $m_0 := 1$, $m_j := 0$ ($j \in V \backslash \{0\}$), $S_0 := 0$, $S_j := -\infty$ ($j \in V \backslash \{0\}$), and $\widetilde{\mathcal{M}} := \mathcal{M}$.

At each step, we do the following: Firstly, we select an activity $j^\star \in \overline{\mathcal{C}}$ to be scheduled by means of an *activity–selection priority rule*. Secondly, we choose a (feasible) mode $\mu^\star \in \widetilde{\mathcal{M}}_{j^\star}$ by means of a *mode–selection priority rule*. Thirdly, we try to find a (feasible) start time $t^\star$ for activity $j^\star$ in mode $\mu^\star$.

In case such a point in time $t^\star$ is found, we schedule activity $j^\star$ in mode $\mu^\star$ at time $t^\star$ and set $\overline{\mathcal{C}} := \overline{\mathcal{C}} \backslash \{j^\star\}$, $\mathcal{C} := \mathcal{C} \cup \{j^\star\}$, $m_j := \mu^\star$, $S_j := t^\star$, and $\widetilde{\mathcal{M}}_j := \{\mu^\star\}$. If we are not able to determine a (feasible) start time $t^\star$, we either select a mode $\mu^{\star\prime} \in \widetilde{\mathcal{M}}_{j^\star}$ with $\mu^{\star\prime} \neq \mu^\star$ and try to schedule activity $j^\star$ in mode $\mu^{\star\prime}$ or we perform *backplanning step 1*. Provided that there does not exist any mode $\mu \in \widetilde{\mathcal{M}}_{j^\star}$ such that activity $j^\star$ can be scheduled in mode μ, *backplanning step 2* is performed.

We select activities out of $\overline{\mathcal{C}}$ until either $\overline{\mathcal{C}}$ is empty, i.e. all activities are scheduled ($\mathcal{C} = V$), or until a prescribed maximum number of backplanning steps has been performed. In the former case, a feasible schedule has been found. In the latter case, the algorithm terminates without having determined a feasible solution.

4.2 Priority rules

In [10] a priority–rule method for RCPSP/max is presented. Here, it is established that, on the average, performing the procedure several times using a different priority rule at each pass leads to better objective function values (but also to higher computational times) than performing the procedure only once with a unique priority rule. The reason is, that the application of one and the same priority rule does not lead to the best objective function value for each problem instance. Therefore, we make use of a multi–pass approach for MRCPSP/max.

For each pass, we need one priority rule for the selection of an activity and one priority rule for the selection of a mode. There are many priority rules that are possible. In particular, the concept of the minimal problem instance $(P^{\widetilde{\mathcal{M}}})$ allows for the application of priority rules that have been proposed in literature for the single–mode case (cf. [1]). In the following, we present three activity– and three mode– selection priority rules. They are used in our procedure because they have turned out to be the most successful ones. Thus, we have nine different combinations of an activity– and a mode–selection rule. We start with the description of the activity–selection rules.

The first activity–selection priority rule ("A1") is referred to as latest start time (LST) rule in literature (cf. [1]). Based on the project network $N^{\widetilde{\mathcal{M}}}$ of the current minimal problem instance $(P^{\widetilde{\mathcal{M}}})$ and the start times of the scheduled activities $l \in \mathcal{C}$, the priority–rule value of an activity $j \in \overline{\mathcal{C}}$ equals its latest start time

$$LS_{j^\star} := \min(\overline{d} - d^{\widetilde{\mathcal{M}}}_{j^\star,n+1}, \min_{l\in\mathcal{C}}(S_l - d^{\widetilde{\mathcal{M}}}_{j^\star l})), \tag{14}$$

where

$$\overline{d} := \sum_{j\in V} \max(\max_{\mu\in\mathcal{M}_j} p_{j_\mu}, \max_{\langle j,l\rangle\in E} \max_{\mu\in\mathcal{M}_j} \max_{\lambda\in\mathcal{M}_l} \delta_{j_\mu l_\lambda}) \tag{15}$$

represents an upper bound on the minimum project duration. When we fix the start time of an activity, the latest (earliest) start times of the nonscheduled activities might decrease (increase). Therefore, it could happen that we cannot find any feasible start time for a nonscheduled activity. In order to detect such an infeasibility early, we assign the highest priority to the activity with the smallest priority–rule value. In this way, we try to save computation time.

The second rule ("A2") is based on $N^{\widetilde{\mathcal{M}}}$, too. The priority–rule value of an activity $j \in \overline{\mathcal{C}}$ equals the difference between its latest start time $LS_{j^\star}$ and its earliest start time

$$ES_{j^\star} := \max(d^{\widetilde{\mathcal{M}}}_{0j^\star}, \max_{l\in\mathcal{C}}(S_l + d^{\widetilde{\mathcal{M}}}_{lj^\star})). \tag{16}$$

This rule is referred to as minimum slack (MSLK) rule in literature (cf. [1]). Since the priority–rule value reflects the maximum number of possible start times of the corresponding activity, its priority increases with decreasing priority–rule value. This is done in order to – like for the LST rule – detect infeasibility early within the scheduling process.

The third rule ("A3") takes into account the current availability of the nonrenewable resources (cf. [14]). For each mode $\mu \in \widetilde{\mathcal{M}}_j$ of each activity $j \in \overline{\mathcal{C}}$ we determine the mean relative consumption

$$\frac{1}{|\mathcal{R}^\nu|} \sum_{k\in\mathcal{R}^\nu} \frac{r^\nu_{j_\mu k}}{R^{\nu\prime}_k} \tag{17}$$

of the nonrenewable resources. $R^{\nu\prime}_k := R^\nu_k - \sum_{l\in\mathcal{C}} r^\nu_{l_{m_l}k}$ stands for the remaining capacity of resource k $(k \in \mathcal{R}^\nu)$. The priority–rule value of an activity $j \in \overline{\mathcal{C}}$ equals the minimum of the values (17) associated with its modes $\mu \in \widetilde{\mathcal{M}}_j$. The activity with the highest priority–rule value receives the highest priority. In this way, we are able to detect infeasibility w.r.t. this type of resources early within the scheduling process.

Next, we present the mode–selection priority rules. The first rule ("M1") is referred to as shortest processing time (SPT) rule in literature (cf. [1]). The priority–rule value of a mode $\mu \in \widetilde{\mathcal{M}}_{j^\star}$ equals $p_{j^\star_\mu}$. The priority of a mode increases with decreasing priority–rule value.

The second rule ("M2") is a modification of a priority rule that is proposed in [10]. For each mode $\mu \in \widetilde{\mathcal{M}}_{j^\star}$ we determine the average weight of all arcs which are incident with activity $j^\star$ in network $N^{\widetilde{\mathcal{M}}}$ under the assumption, that mode μ is chosen. Thus, the priority–rule value equals

$$\frac{1}{|\{\langle j,l\rangle \in E | j = j^\star \vee l = j^\star\}|} \cdot \Big(\sum_{\langle i,j^\star\rangle \in E} \frac{1}{|\widetilde{\mathcal{M}}_i|} \sum_{\iota \in \widetilde{\mathcal{M}}_i} \delta_{i_\iota j^\star_\mu} + \sum_{\langle j^\star,l\rangle \in E} \frac{1}{|\widetilde{\mathcal{M}}_l|} \sum_{\lambda \in \widetilde{\mathcal{M}}_l} \delta_{j^\star_\mu l_\lambda} \Big).$$

The mode with the lowest value receives the highest priority. We expect that the application of the first and the second mode–selection rule leads to rather low objective function values.

The third mode–selection priority rule ("M3") corresponds to the third activity–selection rule. The priority–rule value of a mode $\mu \in \widetilde{\mathcal{M}}_{j^\star}$ equals (17). The highest priority is assigned to the mode with the lowest value.

4.3 Feasibility test

In this subsection, we describe how to check the feasibility of scheduling activity $j^\star$ in mode $\mu^\star$ on the basis of the current partial schedule and the current minimal problem instance. The feasibility test consists of four steps. In step 1, we study the feasibility w.r.t. the nonrenewable resources. In step 2, we check the feasibility w.r.t. the project network of the minimal problem instance. In step 3, we investigate the feasibility w.r.t. the project network of the minimal problem instance where we additionally take into account the start times of the scheduled activities. In step 4, we study the feasibility w.r.t. the renewable resources. If any of these tests fails, we know that the current partial schedule together with activity $j^\star$ scheduled in mode $\mu^\star$ cannot be extended to any feasible schedule. In the following, we give a more detailed description of these steps.

In step 1 (cf. Algorithm 1), we establish whether the choice of mode $\mu^\star$ leads to a *ν–resource conflict*. A ν–resource conflict occurs if the capacity of at least one nonrenewable resource is exceeded by the overall consumption of the activities. For this purpose, we make use of an auxiliary mode set vector $\widetilde{\mathcal{M}}' := \widetilde{\mathcal{M}}$ where we set $\widetilde{\mathcal{M}}'_{j^\star} := \{\mu^\star\}$. $\widetilde{\mathcal{M}}'$ is also used in steps 2 and 3.

Algorithm 1 (Feasibility test step 1)

```
FOR $k \in \mathcal{R}^\nu$ DO
    IF $\sum_{j \in V} \min_{\mu \in \widetilde{\mathcal{M}}'_j} r^\nu_{j_\mu k} > R^\nu_k$ THEN RETURN FALSE
RETURN TRUE
```

Remark 1 The time complexity of Algorithm 1 is $\mathcal{O}(|\mathcal{R}^\nu| \sum_{j \in V} |\mathcal{M}_j|)$.

In step 2 (cf. Algorithm 2), we check whether scheduling activity $j^\star$ in mode $\mu^\star$ leads to a cycle of positive length in network $N^{\widetilde{\mathcal{M}}'}$ of the resulting minimal problem instance $(P^{\widetilde{\mathcal{M}}'})$. This investigation is done by means of the corresponding distance matrix $D^{\widetilde{\mathcal{M}}'}$ which is derived from $D^{\widetilde{\mathcal{M}}}$. A cycle of positive length occurs if we have $d^{\widetilde{\mathcal{M}}'}_{il} + d^{\widetilde{\mathcal{M}}'}_{li} > 0$ for at least one pair $i, l \in V$ of activities.

Algorithm 2 (Feasibility test step 2)

```
$D^{\widetilde{\mathcal{M}}'} := D^{\widetilde{\mathcal{M}}}$
FOR $\langle i, j^\star \rangle \in E$ DO
    $\delta^{\widetilde{\mathcal{M}}'}_{ij^\star} := \min_{\iota \in \widetilde{\mathcal{M}}_i} \delta_{i_\iota j^\star_{\mu^\star}}$ ; Update $D^{\widetilde{\mathcal{M}}'}$
FOR $\langle j^\star, l \rangle \in E$ DO
    $\delta^{\widetilde{\mathcal{M}}'}_{j^\star l} := \min_{\lambda \in \widetilde{\mathcal{M}}_l} \delta_{j^\star_{\mu^\star} l_\lambda}$ ; Update $D^{\widetilde{\mathcal{M}}'}$
IF $\exists\, i, l \in V : d^{\widetilde{\mathcal{M}}'}_{il} + d^{\widetilde{\mathcal{M}}'}_{li} > 0$ THEN RETURN FALSE
RETURN TRUE
```

Remark 2 The time complexity of Algorithm 2 is $\mathcal{O}(|V|^2 \sum_{\langle i, j^\star \rangle \in E} |\mathcal{M}_i| \sum_{\langle j^\star, l \rangle \in E} |\mathcal{M}_l|)$.

In step 3 (cf. Algorithm 3), we compute the earliest start time $ES_{j^\star}$ and the latest start time $LS_{j^\star}$ of activity $j^\star$ based on distance matrix $D^{\widetilde{\mathcal{M}}'}$ and start times S_l of the scheduled activities $l \in \mathcal{C}$. This feasibility test step fails if we establish $ES_{j^\star} > LS_{j^\star}$.

Algorithm 3 (Feasibility test step 3)

```
$ES_{j^\star} := \max(d^{\widetilde{\mathcal{M}}'}_{0j^\star}, \max_{l \in \mathcal{C}}(S_l + d^{\widetilde{\mathcal{M}}'}_{lj^\star}))$
```

```
$LS_{j^\star} := \min(\overline{d} - d^{\widetilde{\mathcal{M}}'}_{j^\star,n+1}, \min_{l \in \mathcal{C}}(S_l - d^{\widetilde{\mathcal{M}}'}_{j^\star l}))$
**IF** $ES_{j^\star} > LS_{j^\star}$ **THEN RETURN** FALSE
**RETURN** TRUE
```

Remark 3 The time complexity of Algorithm 3 is $\mathcal{O}(|V|)$.

In step 4 (cf. Algorithm 4), we establish whether a point in time $t^\star$ exists, such that scheduling activity $j^\star$ in mode $\mu^\star$ at this point in time does not cause a *ϱ–resource conflict*. A ϱ–resource conflict occurs, if there exists a point in time t at which the capacity of at least one renewable resource is exceeded by the usage of the activities being executed at t. Based on the current partial schedule with *partial* mode vector M and *partial* start time vector S, we define a *partial resource profile* $r^\rho_k(\mathcal{C}, M, S, t) := \sum_{j \in \mathcal{A}(\mathcal{C},M,S,t)} r^\rho_{j m_j k}$ for each resource $k \in \mathcal{R}^\rho$ with $\mathcal{A}(\mathcal{C}, M, S, t) := \{j \in \mathcal{C} | S_j \leq t < S_j + p_{j m_j}\}$ denoting the set of scheduled activities being executed at time t. $r^\rho_k(\mathcal{C}, M, S, t)$ reflects the total usage of the set $\mathcal{C}$ of scheduled activities of resource $k \in \mathcal{R}^\rho$ at time t.

For the determination of $t^\star$ we can restrict ourselves to a finite set T of possible start times. A feasible start time of activity $j^\star$ must be included in the interval $[ES_{j^\star}, LS_{j^\star}]$, which is also called *time window*. As we want to minimize the project duration, we try to schedule activity $j^\star$ as early as possible. Therefore, we search for $t^\star$ "from the left to the right" beginning with $ES_{j^\star}$ and continuing at such points in time which equal the completion time of at least one scheduled activity. Hence, we have

$$T := \{ES_{j^\star}\} \cup \{t \in]ES_{j^\star}, \ldots, LS_{j^\star}] \, | t = C_l \wedge l \in \mathcal{C}\}. \tag{18}$$

For a given point in time $t \in T$ we have to see whether $r^\rho_k(\mathcal{C}, M, S, t') + r^\rho_{j^\star_{\mu^\star} k} \leq R^\rho_k$ $(k \in \mathcal{R}^\rho; t' \in T')$ holds. T' denotes the set of points in time for which we have to find out whether a ϱ–resource conflict occurs. Since it is sufficient to take into account the start times of the scheduled activities we have

$$T' := \{t\} \cup \{t' \in] \, t, \ldots, t + p_{j^\star_{\mu^\star}} [\; |t' = S_l \wedge l \in \mathcal{C}\}. \tag{19}$$

Algorithm 4 (Feasibility test step 4)

```
**FOR** $t \in T$ **DO**
    $found :=$ TRUE
    **FOR** $t' \in T'$ **DO**
        **FOR** $k \in \mathcal{R}^\rho$ **DO**
            **IF** $r^\rho_k(\mathcal{C}, M, S, t') + r^\rho_{j^\star_{\mu^\star} k} > R^\rho_k$ **THEN** $found :=$ FALSE
    IF $found$ THEN
        $ES_{j^\star} := t$
        RETURN TRUE
RETURN FALSE
```

Remark 4 The time complexity of Algorithm 4 is $\mathcal{O}(|V|^2|\mathcal{R}^\rho|)$.

4.4 Backplanning

As mentioned in Section 4.1, we differ between two backplanning steps. In the following, these steps are described in detail.

Backplanning step 1 (cf. Algorithm 5) is performed if step 3 or step 4 of the feasibility test fails. The basic idea is, that we try to "enlarge" the *time window* $[ES_{j^\star}, LS_{j^\star}]$ of activity $j^\star$ by increasing $LS_{j^\star}$ such that step 3 or step 4 do no longer fail because of $ES_{j^\star} > LS_{j^\star}$ or $t^\star \not\leq LS_{j^\star}$, respectively.

To increase $LS_{j^\star}$ we have to right–shift some scheduled activities: Firstly, we unschedule each activity $l \in \mathcal{C}$ with $S_l < ES_{j^\star} + d_{j^\star l}^{\widetilde{\mathcal{M}}'}$ by setting $S_l := -\infty$. Let $\overline{\mathcal{C}}'$ denote the set of activities to be unscheduled. Secondly, we increase the earliest start time ES_l of each activity $l \in \overline{\mathcal{C}}'$ to $ES_{j^\star} + d_{j^\star l}^{\widetilde{\mathcal{M}}'}$. Hence, when we (later) reschedule an activity $l \in \overline{\mathcal{C}}'$ at time S'_l, we have $S'_l > S_l$. Note that, if backplanning step 1 is performed after step 4 of the feasibility test, $ES_{j^\star}$ equals the earliest point in time t with $LS_{j^\star} < t \leq \overline{d} - d_{j^\star,n+1}^{\widetilde{\mathcal{M}}'}$, at which activity $j^\star$ can be scheduled in mode $\mu^\star$ without causing a ϱ–resource conflict.

After the unscheduling of $\overline{\mathcal{C}}'$ we have to update the earliest and the latest start time of each activity $l \in \overline{\mathcal{C}}$. If this update yields $ES_l > LS_l$ for at least one activity $l \in \overline{\mathcal{C}}$, the unscheduling of $\overline{\mathcal{C}}'$ is undone and mode $\mu^\star$ is excluded from further consideration. Else, the mode of activity $j^\star$ is fixed to $\mu^\star$ by setting $\widetilde{\mathcal{M}}_{j^\star} := \{\mu^\star\}$.

In order to keep the complexity of our priority–rule method low, the mode m_l of each activity $l \in \overline{\mathcal{C}}'$ remains unchanged. For the same reason, we perform no backplanning but exclude mode $\mu^\star$ from further consideration, if step 1 or step 2 of the feasibility test fails.

Algorithm 5 (Backplanning step 1)

```
$\overline{\mathcal{C}}' := \{l \in \mathcal{C} | S_l < ES_{j^\star} + d_{j^\star l}^{\widetilde{\mathcal{M}}'}\}$ ; $\mathcal{C} := \mathcal{C} \setminus \overline{\mathcal{C}}'$ ; $\overline{\mathcal{C}} := \overline{\mathcal{C}} \cup \overline{\mathcal{C}}'$
FOR $l \in \overline{\mathcal{C}}'$ DO $S_l := -\infty$ ; $ES_l := \max(ES_l, ES_{j^\star} + d_{j^\star l}^{\widetilde{\mathcal{M}}'})$
FOR $l \in \overline{\mathcal{C}}$ DO
```

```
    $ES_l := \max(ES_l, \max_{j \in \mathcal{C}}(S_j + d_{jl}^{\widetilde{\mathcal{M}}'}))$ ; $LS_l := \min(LS_l, \min_{j \in \mathcal{C}}(S_j - d_{lj}^{\widetilde{\mathcal{M}}'}))$
IF $\exists\, l \in \overline{\mathcal{C}} : ES_l > LS_l$ THEN
    $\overline{\mathcal{C}} := \overline{\mathcal{C}} \setminus \overline{\mathcal{C}}'$ ; $\mathcal{C} := \mathcal{C} \cup \overline{\mathcal{C}}'$ ; $\widetilde{\mathcal{M}}_{j^\star} := \widetilde{\mathcal{M}}_{j^\star} \setminus \{\mu^\star\}$
    FOR $l \in \overline{\mathcal{C}}$ DO Update $ES_l, LS_l$
ELSE $\widetilde{\mathcal{M}}_{j^\star} := \{\mu^\star\}$
RETURN
```

Remark 5 The time complexity of Algorithm 5 is $\mathcal{O}(|V|^2)$.

Backplanning step 2 (cf. Algorithm 6) is performed if activity $j^\star$ cannot be scheduled in any mode $\mu \in \widetilde{\mathcal{M}}_{j^\star}$. In this case, we unschedule the previously selected activity $j^{\star\prime}$ and prevent the rescheduling of $j^{\star\prime}$ for a certain number of scheduling steps.

Algorithm 6 (Backplanning step 2)

```
Determine $j^{\star\prime}$ ; $\mathcal{C} := \mathcal{C} \setminus \{j^{\star\prime}\}$ ; $\overline{\mathcal{C}} := \overline{\mathcal{C}} \cup \{j^{\star\prime}\}$ ;
$m_{j^{\star\prime}} := 0$ ; $S_{j^{\star\prime}} := -\infty$ ; $\widetilde{\mathcal{M}}_{j^{\star\prime}} := \mathcal{M}_{j^{\star\prime}}$; Determine $D^{\widetilde{\mathcal{M}}}$
FOR $l \in \overline{\mathcal{C}}$ DO Update $ES_l, LS_l$
RETURN
```

Remark 6 The time complexity of Algorithm 6 is $\mathcal{O}(|V|^3)$.

4.5 Algorithm

In this subsection, we give a rough pseudo–code description of the priority–rule method (cf. Algorithm 7).

Algorithm 7 (Priority–rule method)

```
$\mathcal{C} := \{0\}$ ; $\overline{\mathcal{C}} := V \setminus \{0\}$ ; $(M, S) := ((1, 0, \ldots, 0), (0, -\infty, \ldots, -\infty))$
$\widetilde{\mathcal{M}} := \mathcal{M}$ ; Determine $D^{\widetilde{\mathcal{M}}}$ and $\overline{d}$ ; $ES := (d_{j0}^{\widetilde{\mathcal{M}}})_{j \in V}$ ; $LS := (\overline{d} - d_{j,n+1}^{\widetilde{\mathcal{M}}})_{j \in V}$
WHILE $\overline{\mathcal{C}} \neq \emptyset$ DO
    Select $j^\star \in \overline{\mathcal{C}}$ ; $found :=$ FALSE
    WHILE $\widetilde{\mathcal{M}}_{j^\star} \neq \emptyset$ AND NOT $found$ DO
        Select $\mu^\star \in \widetilde{\mathcal{M}}_{j^\star}$
        IF Algorithm 1 = TRUE AND Algorithm 2 = TRUE THEN
            IF Algorithm 3 = TRUE AND Algorithm 4 = TRUE THEN
                $found$ := TRUE ; $\overline{\mathcal{C}} := \overline{\mathcal{C}} \setminus \{j^\star\}$ ; $\mathcal{C} := \mathcal{C} \cup \{j^\star\}$
                $m_{j^\star} := \mu^\star$ ; $S_{j^\star} := ES_{j^\star}$ ; $\widetilde{\mathcal{M}}_{j^\star} := \{\mu^\star\}$ ; Update $D^{\widetilde{\mathcal{M}}}$
                FOR $l \in \overline{\mathcal{C}}$ DO Update $ES_l$, $LS_l$
            ELSE Algorithm 5
        ELSE $\widetilde{\mathcal{M}}_{j^\star} := \widetilde{\mathcal{M}}_{j^\star} \setminus \{\mu^\star\}$
    IF $\widetilde{\mathcal{M}}_{j^\star} = \emptyset$ THEN Algorithm 6
RETURN $(M, S)$
```

Since one pass of Algorithm 7 takes only a little amount of time (approximately 0.1 seconds for problem instances with 100 nondummy activities) we extend the multi–pass approach by embedding it into *regret–based biased random sampling* (cf. [7], [8]): The execution of Algorithm 7 is repeated until a prescribed time limit is reached. At each pass, we use a combination of an activity–selection and a mode–selection priority rule. (Hence, each combination is used again after eight passes.) At this, we determine activity $j^\star$ and mode $\mu^\star$ not in a deterministic but in a stochastic manner by using the priority–rule values as a basis for a *roulette selection*. This means that the probability of the selection of an activity or a mode with high priority is higher than the probability of the selection of an activity or a mode with low priority, respectively.

If a pass of Algorithm 7 returns a feasible schedule (M, S), S_{n+1} can be used as a tighter upper bound $\overline{d}$ for the following passes. The result of the sampling procedure equals the schedule with the lowest project duration among all feasible schedules which have been returned by Algorithm 7.

5 Computational results

In this section, we evaluate the performance of our priority–rule method. In Section 5.1, we describe the benchmark test set which is made use of in the following subsections. The impact of the available computation time and certain control parameters is analyzed in Sections 5.2 and 5.3, respectively. In Sections 5.4 and 5.5, we investigate the effect of the priority rules and the backplanning feature, respectively. Finally, the priority–rule method is compared with the other existing heuristic solution procedures for MR-CPSP/max (cf. Section 5.6).

Table 1. Parameter setting

Parameter	Value(s)
n	100
m	3, 4, 5
$\lvert\mathcal{R}^\rho\rvert$	3
$\lvert\mathcal{R}^\nu\rvert$	3
RT	0.50
RF^ϱ	1.00
RF^ν	1.00
RS^ϱ	0.25, 0.50, 0.75
RS^ν	0.25, 0.50, 0.75

5.1 Benchmark problem instances

In order to evaluate the performance of the priority–rule method a test set has been generated using the problem generator ProGen/max (cf. [23]). The most important control parameters used for the full factorial design are given by Table 1. In an attempt to keep the size of the experiment small, certain parameters have been fixed at specific values (according to [4]). For each parameter combination, 10 problem instances have been generated. Hence, this test set consists of 270 MRCPSP/max instances.

In the following, we explain the meaning of the control parameters. For a more detailed description we refer to [16], [17], [22], and [23]:

- n denotes the number of nondummy activities of the project.
- m stands for the number of modes which are available for each activity $j \in \{1, \dots, n\}$.
- The restrictiveness RT (cf. [23]) reflects the degree of parallelity of the corresponding network without cycles. With increasing degree of parallelity, RT decreases.
- The resource factor RF (cf. [21]) corresponds to the average percentage of resources required per activity. RF is defined for renewable resources (RF^ϱ) as well as for nonrenewable resources (RF^ν).
- The resource strength RS (cf. [14]) reflects the ratio of resource capacity and resource demand and is defined for renewable resources (RS^ϱ) as well as for nonrenewable resources (RS^ν).

The parameters RT, RF^ϱ, RF^ν, RS^ϱ, and RS^ν are normalized to the interval [0,1].

Next, we describe the impact of these parameters on the size and the difficulty of a problem instance (cf. [22]): The size of a problem instance is influenced by the parameters n, m, $|\mathcal{R}^\rho|$, and $|\mathcal{R}^\nu|$. It increases with increasing parameter values. The parameters RT, RF^ϱ, RF^ν, RS^ϱ, and

Table 2. Feasible

m	10 sec.	100 sec.
3	100	100
4	100	100
5	100	100

Table 3. Deviation

m	10 sec.	100 sec.
3	113	97
4	178	156
5	256	221

RS^ν reflect the difficulty of a problem instance: According to [22], difficult instances are characterized by a high degree of parallelity (i.e. a low value of RT) and high resource factors RF^ϱ and RF^ν. In [4], it is reported that the influence of RS^ϱ and RS^ν on the difficulty can be characterized by a bell–shaped curve and a U–shaped curve, respectively.

The full factorial analysis has been done on an Intel Pentium III 800 MHz personal computer with 256 MB SDRAM under the operating system Windows 2000. All algorithms have been coded in ANSI C using the MS Visual C++ 6.0 Developer Studio.

5.2 Impact of time bounds

In this subsection, we test the performance of our priority–rule method ("Prio") w.r.t. two time bounds: 10 seconds and 100 seconds. In Table 2, we give for each value of m and for each time bound the percentage of problem instances of the test set for which a feasible solution has been determined by "Prio" within the given amount of computation time. Within 10 seconds, all instances have already been solved to feasibility.

In Table 3, we give the average percentage deviation of the objective function value from a lower bound value on the minimum project duration (cf. [12]). It can be seen, that, as expected, the deviation increases with increasing value of m. If the available amount of computation time is increased from 10 seconds to 100 seconds, the average percentage deviation for the whole test set decreases from 182% to 158%.

5.3 Impact of control parameters

The most important control parameters which have been varied are m, RS^ϱ, and RS^ν. In the following, we try to find out the influence of the latter two parameters on the performance of "Prio". The corresponding deviations from a lower bound value (for a time bound of 10 seconds) can be found in Table 4. Here, each row (column) is associated with a certain value of RS^ϱ (RS^ν). For "Prio", the most difficult instances are those with a medium value of RS^ϱ and a low value of RS^ν. The entries of Table 4 seem to support the

Table 4. Control parameters

$RS^{\varrho\backslash\nu}$	0.25	0.50	0.75
0.25	287	134	122
0.50	291	135	140
0.75	265	134	135

statement given in [4] that the influence of RS^{ϱ} (RS^{ν}) on the difficulty of a problem instance can be described by means of a bell–shaped (U–shaped) curve.

5.4 Effect of priority rules

In this subsection, we analyze the impact of the priority rules on the performance of “Prio” (w.r.t. a time bound of 10 seconds). For this, we firstly compare “Prio” with a version, named “Rand”, where all priority rules have been replaced by a simple random priority rule. Secondly, we investigate the effect of the priority rules by means of a comparison with “Rand”.

In Table 5, we give for each value of m the percentage of problem instances for which a feasible solution has been determined within the given amount of time. “Rand” determines for only 85% of the instances of the test set a feasible solution. In Table 6, we give the average percentage deviation of the objective function value from a lower bound value on the minimum project duration for those instances for which a feasible solution has been found by “Prio” and “Rand’ ’. It turns out that, for the whole test set, the deviation w.r.t. “Prio” (162%) is much smaller than the deviation w.r.t. “Rand” (311%).

Next, we try to find out the effect of the individual priority rules. For this, for each priority rule, we have implemented a version of “Rand” which differs in the sense that the respective rule replaces the corresponding simple random priority rule. The resulting versions are named according to the priority rules (“A1”, “A2”, “A3”, “M1”, “M2”, and “M3”) and compared with “Rand”. From Table 5, we can conclude that – w.r.t. the criterion “feasibility” – the most effective activity- and mode-selection priority rules are “A2” and “M3”, respectively. The corresponding values for the whole test set equal 87% (for “A2”) and 99% (for “M3”).

From Table 6 we can draw the conclusion that – w.r.t. the criterion “deviation from a lower bound value” – the most effective activity– and mode–selection priority rules are “A1” and “M1”, respectively. The corresponding values for the whole test set equal 306% (for “A1”) and 170% (for “M1”). Note, that the deviations are based on those instances which have been solved to feasibility by “Rand” and the respective version of “Rand”.

Table 5. Feasible – Priority rules

m	Prio	Rand	A1	A2	A3	M1	M2	M3
3	100	79	79	81	74	79	80	99
4	100	88	91	91	88	88	89	99
5	100	88	87	89	87	87	89	100

Table 6. Deviation – Priority rules

m	Prio	Rand	A1	A2	A3	M1	M2	M3
3	89	180	178	178	180	95	96	177
4	159	305	300	303	303	171	184	297
5	229	435	427	431	429	235	270	422

Table 7. Feas. – Backplanning

m	Prio	B0
3	100	31
4	100	37
5	100	37

Table 8. Dev. – Backplanning

m	Prio	B0
3	101	111
4	165	183
5	253	262

Summing up it may be concluded that the strength of “Prio” lies especially in the combination of all priority rules which leads to a superior performance compared to “Rand”.

5.5 Effect of backplanning

The impact of the backplanning procedures on the performance of “Prio” is analyzed in the following. For this, we have implemented a version without backplanning, named “B0”. As can be seen from Table 7, the percentage of instances solved to feasibility without the backplanning feature is very small: Only 35% of the instances of the test set could have been solved to feasibility by “B0”. In Table 8, we give the average percentage deviations from a lower bound value for those instances for which a feasible solution has been found by “Prio” and “B0”. For the whole test set, we have 177% for “Prio” and 189% for “B0”.

All in all, it can be said that the backplanning feature is an essential part of “Prio”. In particular, this holds for the criterion “feasibility”.

Table 9. Feas. – 10 sec.

m	TS_{DR}	TS_F	Prio
3	28	39	100
4	33	46	100
5	42	59	100

Table 10. Dev. – 10 sec.

m	TS_{DR}	TS_F	Prio
3	151	117	80
4	286	171	131
5	433	229	200

Table 11. Feas. – 100 sec.

m	TS_{DR}	TS_F	Prio
3	53	39	100
4	61	46	100
5	67	59	100

Table 12. Dev. – 100 sec.

m	TS_{DR}	TS_F	Prio
3	40	105	63
4	91	151	113
5	164	215	170

5.6 Comparison with other heuristics

In this subsection, we compare “Prio” with the heuristics of [4] (“TS_{DR}”) and [10] (“TS_F”), which were reimplemented. (Details of the parameter settings of “TS_{DR}” and “TS_F” can be found in [5] and [11], respectively.) Two time bounds – 10 seconds and 100 seconds – have been given. The evaluation criteria are the percen tage of problem instances for which a feasible solution has been determined by the respective heuristic and the average percentage deviation of the objective function value from a lower bound value for those instances for which a feasible solution has been determined by all heuristics within the given amount of time.

For a time bound of 10 seconds (cf. Tables 9 and 10), the best results are obtained by “Prio”. Here, as has already been shown in Section 5.2, for all instances of the test set a feasible solution has been found and the average deviation for the whole test set equals 146%. “TS_{DR}” determines for only 34% and “TS_F” for only 48% of the instances of the test set a feasible solution. The average deviation from a lower bound value equals 309% and 180%, respectively.

For a time bound of 100 seconds (cf. Tables 11 and 12), the results (for the whole test set) are as follows: The percentage of instances for which a feasible solution has been determined by “TS_{DR}” increases from 34% to 60%, and the average deviation decreases from 309% to 108%, which is smaller than the corresponding value obtained by “Prio” (124%). “TS_F” is not able to solve more instances to feasibility within 100 seconds than within 10 seconds. The corresponding average percentage deviation equals 165%. All in all, we can conclude that especially “TS_{DR}” takes advantage of the additional amount of available computation time.

6 Conclusions

In this paper, a heuristic for MRCPSP/max has been presented. This priority–rule method with backplanning relies on an integration approach for MRCPSP/max which is adapted from that for MRCPSP. For this approach, a special relaxation has been presented, which allows for the application of priority rules for the single–mode case and of a very effectice feasibility test. The heuristic is embedded in random sampling thereby using different combinations of priority rules.

The computational performance analysis has shown that the priority–rule method is able to determine a feasible solution for each instance of the test set whereas the existing solution procedures have difficulties in this regard. It is essential that a procedure, which is applied in practice, returns a feasible solution within a small amount of time.

Areas of further research are e.g. the improvement of the priority–rule method w.r.t. optimality and its adaptation to resource–constrained project scheduling problems with a different objective function.

References

1. Alvarez–Valdes R, Tamarit JM (1989) Heuristic algorithms for resource–constrained project scheduling: a review and empirical analysis. In: Słowiński R, Węglarz J (eds) Advances in project scheduling, pp 113–134. Elsevier Science, Amsterdam
2. Bartusch M, Möhring R, Radermacher FJ (1988) Scheduling project networks with resource constraints and time windows. Annals of Operations Research 16: 201–240
3. Brucker P, Drexl A, Möhring R, Neumann K, Pesch E (1999) Resource–constrained project scheduling: notation, classification, models, and methods. European Journal of Operational Research 112: 3–41
4. De Reyck B (1998) Scheduling projects with generalized precedence relations: exact and heuristic procedures. PhD thesis, Katholieke Universiteit Leuven, Belgium
5. De Reyck B, Herroelen W (1997) The multi–mode resource–constrained project scheduling problem with generalized precedence relations. Onderzoeksrapport 9725, Katholieke Universiteit Leuven, Belgium
6. Dorndorf U, Pesch E, Toàn PH (1998) A time–oriented branch–and–bound algorithm for resource constrained project scheduling with generalized precedence constraints. Working paper, Institut für Wirtschafts– und Gesellschaftswissenschaften, University of Bonn
7. Drexl A (1991) Scheduling of project networks by job assignment. Management Science 37: 1590–1602
8. Drexl A, Grünewald J (1993) Nonpreemptive multi–mode resource–constrained project scheduling. IEE Transactions 25: 74–81
9. Elmaghraby SE (1977) Activity networks – project planning and control by network models. Wiley, New York
10. Franck B (1999) Prioritätsregelverfahren für die ressourcenbeschränkte Projektplanung mit und ohne Kalender. PhD thesis, Shaker, Aachen

11. Franck B, Schäfer S, Seidel K (1998) Heuristiken für das Multi–Mode Project Scheduling Problem. Working paper, Institute of Economic Theory and Operations Research, University of Karlsruhe
12. Heilmann R (2000) A branch–and–bound procedure for the multi–mode resource–constrained project scheduling problem with minimum and maximum time lags. Report WIOR–578, University of Karlsruhe (*submitted to:* European Journal of Operational Research)
13. Herroelen W, Demeulemeester E, De Reyck B (1999) A classification scheme for project scheduling. In: Węglarz J (ed): Project scheduling: recent models, algorithms and applications, pp 1–26. Kluwer, Boston
14. Kolisch R (1995) Project scheduling under resource constraints: efficient heuristics for several problem classes. Physica, Heidelberg
15. Kolisch R, Drexl A (1997) Local search for nonpreemptive multi–mode resource-constrained project scheduling. IIE Transactions 29: 987–999
16. Kolisch R, Schwindt C, Sprecher A (1999) Benchmark instances for project scheduling problems. In: Węglarz J (ed) Project scheduling: recent models, algorithms and applications, pp 197–212. Kluwer, Boston
17. Kolisch R, Sprecher A, Drexl A (1995) Characterization and generation of a general class of resource–constrained project scheduling problems. Management Science 41: 1693-1703
18. Lawler, EL (1976) Combinatorial optimization: networks and matroids. Holt, Rinehart, and Winston, New York
19. Möhring R, Stork F, Uetz M (1998) Resource constrained project scheduling with time windows: a branching scheme based on dynamic release dates. Working paper 596, Fachbereich Mathematik, Technical University of Berlin
20. Neumann K, Morlock M (1993) Operations research. Hanser, München
21. Pascoe TL (1966) Allocation of resources C.P.M. Revue Française Recherche Opérationnelle 38: 31–38
22. Schwindt C (1998a) Verfahren zur Lösung des ressourcenbeschränkten Projektdauerminimierungsproblems mit planungsabhängigen Zeitfenstern. PhD thesis, Shaker, Aachen
23. Schwindt C (1998b) Generation of resource–constrained project scheduling problems subject to temporal constraints. Report WIOR–543, University of Karlsruhe
24. Słowiński R, Soniewicki B, Węglarz J (1994) DSS for multiobjective project scheduling. European Journal of Operational Research 79: 220–229
25. Sprecher A (1994) Resource–constrained project scheduling – exact methods for the multi–mode case. Berlin, Heidelberg, New York: Springer
26. Sprecher A, Drexl A (1998) Multi–mode resource–constrained project scheduling by a simple, general and powerful sequencing algorithm. European Journal of Operational Research 107: 431–450
27. Sprecher A, Hartmann S, Drexl A (1997) An exact algorithm for project scheduling with multiple modes. OR Spektrum 19: 195–203
28. Van Hove JC, Deckro R (1998) Multi–modal project scheduling with generalized precedence constraints. Proceedings of the PMS Workshop: 137–140