Sprecher, Arno; Drexl, Andreas

Solving Multi-Mode Resource-Constrained Project Scheduling Problems by a Simple, General and Powerful Sequeacing Algorithm. Part I: Theory

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 385

No. 385

Solving Multi-Mode Resource-Constrained Project Scheduling Problems by a Simple, General and Powerful Sequencing Algorithm. Part I: Theory

Arno Sprecher and Andreas Drexl

January 1996

Arno Sprecher, Andreas Drexl
Institut für Betriebswirtschaftslehre, Christian-Albrechts-Universität zu Kiel,
Olshausenstr. 40, 24098 Kiel, Germany
Tel.&Fax ++49(0)431-880-1531, Email: Drexl@bwl.uni-kiel.de.

Abstract

In this paper we present an exact solution procedure of the branch-and-bound type for solving the muti-mode resource-constrained project scheduling problem. The simplicity of the enumeration scheme enables a compact representation of the current state of the search process. This representation can be employed to formulate and prove search tree reduction schemes which highly increase the performance of the algorithm. Among the benefits of the approach are ease of description, ease of implementation, ease of generalization, and, additionally, superior performance of the exact approach as well as reasonable heuristic capabilities of the truncated method.

The procedure has been coded in C and implemented on a personal computer as well as on a workstation. In the second part of the paper we present the results of our experimental investigations. The experimental results obtained by using the standard project generator ProGen demonstrate a superior performance of the exact and heuristic approach. The size of the instances that can be solved to optimality has been nearly doubled.

Keywords: Project Scheduling, Resource Constraints, Multiple Modes, Branch-and-Bound, Heuristic, Computational Results.

1 Introduction

Whereas the standard methods of project scheduling, CPM and MPM, base on the assumption of unlimited availability of the resources involved, the modern concepts include more realistic aspects. In general, the availabilities of the resources involved are limited.

Consequently, numerous publications have dealt with exact methods for solving the so-called single-mode resource-constrained project scheduling problem (SRCPSP) (cf. e.g. [4], [5], [6], [17], [27]), where each of the activities comprising the project has to be performed in one prescribed way (mode) using certain amounts of the resources provided. The objective considered is, as performed by CPM and MPM, the minimization of the makespan. Recent advances have incorporated a little more of reality. The activities can be executed in one out of several modes. The modes reflect alternative combinations of resources and belonging quantities employed to fulfill the tasks related to the activities. The activity duration is a discrete function of the employed quantities, that is, using this concept e.g. working-off an activity can be accelerated by raising the quantities coming into operation (time-resource-tradeoff). Moreover, by raising the quantities of some resources and reducing the quantities of others the resource substitution (resource-resource-tradeoff) can be realized. The problem at hand is the multi-mode resource constrained project scheduling problem (MRCPSP), which is commonly considered with makespan minimization as objective.

Using the categorization scheme proposed by Slowinski (cf. [19], [20]) and Weglarz (cf. [31], [32]) we distinguish three categories of resources required for the execution of the project. Namely, renewable, nonrenewable and doubly constrained resources.

Renewable resources are available on a period-by-period basis, that is, the quantities available are renewed from period to period (hour, day, week, month). The per-period availability may be constant or vary from period to period. Manpower, machines, fuelflow and space are renewable resources.

In contrast to the renewable resources, nonrenewable ones are limited on a total project basis, that is, instead of a limited per-period usage of the renewable resources we have a limited overall consumption of nonrenewable resources for the entire project. Money, energy and raw material belong to this category.

Resources which are limited on total project basis as well as on per-period basis are called doubly constrained. Money represents a resource of this type if beside the total project expenditures the per-period cashflow is limited. Manpower can be a doubly constrained resource, too, if for example a skilled worker can only spend a limited number of periods on the project.

Whereas the exact methods for the single-mode problem are well documented in the literature the multi-mode problem has attracted less attention (cf. [12], [15], [16], [23], [25], [28], and [29]). We will present an enumeration scheme for optimally solving the multi-mode resource-constrained project scheduling problem (MRCPSP) with makespan minimization as objective. The basic scheme is generalized for dealing with any regular measure of performance and enhanced by powerful acceleration schemes. The approach considerably extends the precedence tree guided enumeration scheme proposed by Patterson et al. (cf. [15], [16]). The benefits of the approach are (1) ease of description, (2) ease of implementation, (3) ease of generalization, (4) superior performance of the exact method and (5) reasonable heuristic capabilities of the truncated method.

The paper is organized as follows: Section 2 is devoted to the detailed description of the model under consideration. Section 3 addresses the basic branch-and-bound procedure. Section 4 presents search tree reduction techniques and Section 5 draws the conclusions. In the second part of the paper we will present the results of our thorough computational study.

2 The Model

We consider a project which consists of J activities (jobs, tasks). Due to technological requirements precedence relations between some of the jobs enforce that a job j, $j = 2, \ldots, J$, may not be started before all its predecessors h, $h \in \mathcal{P}_j$, are finished. The structure of the project is depicted by a so-

called activity-on-node (AON) network where the nodes and the arcs represent the jobs and precedence relations, respectively. The network is acyclic and numerically labelled, that is an activity j has always a higher number than all its predecessors. W.o.l.o.g. activity 1 is the only start activity (source) and activity J is the only finish activity (sink).

Each activity j, $j = 1,\ldots,J$, may be executed in one out of M_j modes. The activities may not be preempted and a mode once selected may not change, i.e. a job j once started in mode m has to be completed in mode m without interruption. Performing job j in mode m takes d_{jm} periods and is supported by a set R of renewable, a set N of nonrenewable and a set D of doubly constrained resources. Considering a horizon, that is, an upper bound $\overline{T}$ on the project's makespan, we have an available amount of K^{ρ}_{rt} (K^{δ}_{rt}) units of renewable (doubly constrained) resource r, $r \in R$ ($r \in D$), in period t, $t = 1,\ldots,\overline{T}$. The overall capacity of the nonrenewable resource r, $r \in N$, and doubly constrained resource r, $r \in D$, is given by K^{ν}_{r} and K^{δ}_{r}, respectively. If job j is scheduled in mode m then k^{ρ}_{jmr} units of renewable resource r, $r \in R$, are used and k^{δ}_{jmr} units of doubly constrained resource r, $r \in D$, are consumed each period job j is in process. Additionally, k^{ν}_{jmr} units of nonrenewable resource r, $r \in N$, are consumed. The parameters are summarized in Table 1 and assumed as integer-valued. The objective is to find a makespan minimal schedule that meets the constraints imposed by the precedence relations and the limited resource availabilities. Clearly, since the doubly constrained resources can easily be taken into account by appropriately enlarging the sets of renewable and nonrenewble resources they do not have to be considered explicitly.

If $|N| > 0$ then finding a feasible solution is an NP-hard problem (*cf.* [13]). However, presuming feasibility and a constant per-period availability of the renewable resources, an upper bound on the minimum makespan is given by the sum of maximum activity durations.

Given an upper bound $\overline{T}$ on the project's makespan we can use the precedence relations and the modes of shortest duration to derive time windows, i.e. intervals $[EF_j, LF_j]$, with earliest finish time EF_j and latest finish time LF_j, containing the precedence feasible completion times of activity j, $j = 1,\ldots,J$, by traditional forward and backward recursion as performed in MPM. Analogously, the interval $[ES_j, LS_j]$ bounded from below and above by the earliest start time ES_j and the latest start time LS_j, respectively, can be calculated to reflect the precedence feasible start times. The benefit is twofold: First, the number of variables used in the integer (binary) programming formulation is reduced substantially. Second, within a branch-and-bound algorithm the bounds can be efficiently used to speed up the convergence.

Note, since different modes may have different durations, starting an activity j within the time window $[ES_j, LS_j]$ does not necessarily mean that the job is completed in the interval $[EF_j, LF_j]$.

J	:	number of jobs
M_j	:	number of modes job j can be performed in
d_{jm}	:	duration of job j being performed in mode m
$R\ (N)$	:	set of renewable (nonrenewable) resources
$\overline{T}$	:	upper bound on the project's makespan
$K_{rt}^{\rho} \geq 0$	:	number of units of renewable resource r, $r \in R$, available in period t, $t = 1, \ldots, \overline{T}$
$K_r^{\nu} \geq 0$	:	total number of units available of nonrenewable resource r, $r \in N$
$k_{jmr}^{\rho} \geq 0$	:	number of units of renewable resource r, $r \in R$, used by job j being performed in mode m each period the job is in process
$k_{jmr}^{\nu} \geq 0$	:	number of units of nonrenewable resource r, $r \in N$, consumed by job j being performed in mode m
$\mathcal{P}_j\ (\mathcal{S}_j)$	:	set of immediate predecessors (successors) of job j
$ES_j\ (EF_j)$	:	earliest start time (finish time) of job j, calculated by using minimal job durations and neglecting resource usage (consumption)
$LS_j\ (LF_j)$	:	latest start time (finish time) of job j, calculated by using minimal job durations, neglecting resource usage (consumption) and taking into account the upper bound $\overline{T}$ on the project's duration

Table 1: Symbols and Definitions

Using the time windows derived we can now state the problem as a linear program. It was similarly presented by Talbot (cf. [29]). We use binary decision variables x_{jmt}, $j = 1, \ldots, J$, $m = 1, \ldots, M_j$, $t = EF_j, \ldots, LF_j$,

$$x_{jmt} = \begin{cases} 1 & , \text{ if job } j \text{ is performed in mode } m \text{ and completed at the end of period } t \\ 0 & , \text{ otherwise.} \end{cases}$$

The model is presented in Table 2 and referred to as the multi-mode resource-constrained project scheduling problem (MRCPSP).

Since there is exactly one finish activity, the objective function (1) realizes the minimization of the project's makespan. Constraints (2) ensure that exactly one mode and one completion time is assigned to each activity. The precedence relations are taken into account by (3). (4) guarantees, that the per-period availabilities of the renewable resources are not exceeded. Finally, (5) secures feasibility with respect to the consumable (nonrenewable) resources.

Obviously, the well-known flow-shop, job-shop-, open-shop and assembly line balancing problem are included in the model outlined above (cf. e.g. [24], pp. 10). Thus, the problem is a member of the class of NP-hard problems (cf. [11]).

$$\text{Minimize } \Phi(x) = \sum_{m=1}^{M_J} \sum_{t=EF_J}^{LF_J} t \cdot x_{Jmt} \tag{1}$$

s.t.

$$\sum_{m=1}^{M_j} \sum_{t=EF_j}^{LF_j} x_{jmt} = 1 \qquad j = 1, \ldots, J \tag{2}$$

$$\sum_{m=1}^{M_h} \sum_{t=EF_h}^{LF_h} t \cdot x_{hmt} \leq \sum_{m=1}^{M_j} \sum_{t=EF_j}^{LF_j} (t - d_{jm}) x_{jmt} \qquad j = 2, \ldots, J, h \in \mathcal{P}_j \tag{3}$$

$$\sum_{j=1}^{J} \sum_{m=1}^{M_j} k_{jmr}^{\rho} \sum_{q=\max\{t,EF_j\}}^{\min\{t+d_{jm}-1,LF_j\}} x_{jmq} \leq K_{rt}^{\rho} \qquad r \in R, t = 1, \ldots, \overline{T} \tag{4}$$

$$\sum_{j=1}^{J} \sum_{m=1}^{M_j} k_{jmr}^{\nu} \sum_{t=EF_j}^{LF_j} x_{jmt} \leq K_r^{\nu} \qquad r \in N \tag{5}$$

$$x_{jmt} \in \{0, 1\} \qquad j = 1, \ldots, J, m = 1, \ldots, M_j, \quad t = EF_j, \ldots, LF_j \tag{6}$$

Table 2: The Model of the MRCPSP

Moreover, the model presented above can be easily modified to include time-varying request (cf. [8]) and generalized temporal constraints (cf. [2], [24], pp. 19). However, note, if negative minimal time-lags are incorporated then the objective function has to adapted to reflect the minimization of the makespan. Additionally, further objectives can be considered. For the later representation of the algorithm it will pay to classify the objectives: We extend the definition of a regular performance measure given in [1], [10], [17] and [18] to the multi-mode case (cf. [24], pp. 22):

Definition 1

Let $\mathcal{M} := \{1, \ldots, M_1\} \times \cdots \times \{1, \ldots, M_J\}$ *and* $CT_1, \ldots, CT_J$ *be the completion times of job 1,..., job J scheduled in mode* $m_1, \ldots, m_J$*, respectively. A performance measure* Φ *is a mapping*

$$\Phi : \mathbf{Z}_{\geq 0}^{J} \times \mathcal{M} \longrightarrow \mathbf{R}_{\geq 0}$$

which assigns a performance value $\Phi(CT, M)$ *to each pair of a J-tuple* $CT = (CT_1, \ldots, CT_J)$ *of completion times and* $M = (m_1, \ldots, m_J)$ *of modes. If* Φ *is monotonically increasing with respect to (the componentwise ordering of* $\mathbf{Z}_{\geq 0}^{J}$ *of) the first component, that is,*

$$\Phi((CT_1, \ldots, CT_J), M) \;<\; \Phi((\overline{CT}_1, \ldots, \overline{CT}_J), M)$$

implies

$$CT_j \;<\; \overline{CT}_j$$

for at least one j, $j \in \{1,\ldots,J\}$, and additionaly minimization of Φ is considered, then we call the performance measure regular.

Clearly, we can denote Φ as a function of the binary variables introduced, therefore, loosely in notation we yield the following regular measures of performance: Let ρ_j ($\overline{\delta}_j$) denote the release date (due date) of activity j, and let c_{jmt} be the cashflow induced by job j being performed in mode m and completed in period t, $j = 1,\ldots,J$, $m = 1,\ldots,M_j$, $t = EF_j,\ldots,LF_j$.

(a) The minimization of the project's makespan

$$\Phi(CT,M) \;:=\; \sum_{m=1}^{M_J}\sum_{t=EF_J}^{LF_J} t\cdot x_{Jmt} = CT_J$$

(b) The minimization of the weighted delays

$$\Phi(CT,M) \;:=\; \frac{1}{J}\sum_{j=1}^{J}\sum_{m=1}^{M_j}\sum_{t=\overline{\delta}_j+1}^{LF_j}(t-\overline{\delta}_j)\cdot x_{jmt} = \frac{1}{J}\sum_{j=1}^{J}(CT_j-\overline{\delta}_j)$$

(c) The minimization of the total number of tardy activities

$$\Phi(CT,M) \;:=\; \sum_{j=1}^{J}\sum_{m=1}^{M_j}\sum_{t=\overline{\delta}_j+1}^{LF_j} x_{jmt} = |\{j\in\{1,\ldots,J\}; CT_j > \overline{\delta}_j\}|$$

(d) The minimization of the mean weighted flow time

$$\Phi(CT,M) \;:=\; \frac{1}{J}\sum_{j=1}^{J}\sum_{m=1}^{M_j}\sum_{t=EF_j}^{LF_j}(t-\rho_j)\cdot x_{jmt} = \frac{1}{J}\sum_{j=1}^{J}(CT_j-\rho_j)$$

(e) The maximization of the net present value

$$\Phi(CT,M) \;:=\; \sum_{j=1}^{J}\sum_{m=1}^{M_j}\sum_{t=EF_j}^{LF_j} c_{jmt}\, x_{jmt}$$

Note, more precisely, the net-present value is a regular measure of performance if $c_{jmt} \geq c_{jm,t+1}$ for $j = 1,\ldots,J$, $m = 1,\ldots,M_j$, $t = EF_j,\ldots,LF_j - 1$, holds and the objective function is multiplied by -1.

Additional regular performance measures as e.g. the total (weighted) resource consumption and non-regular performance measures, as e.g. the smoothness of the resource profile can be found in [21] and [22].

The generalized problem we obtain by replacing the makespan minimization by any regular measure of performance is referred to as the generalized resource-constrained project scheduling problem (GRCPSP).

3 The Branch-and-Bound Algorithm

In this section we present and analyze an algorithm for the MRCPSP. The foundation stone of the basic algorithm is laid through a search tree reduction proposed by Patterson et al. (cf. [15], [16]). By implicitly using dominating start time assignments, they made a major contribution to improve earlier versions (cf. [29], [30]) to a precedence tree guided enumeration scheme. The algorithm is completely restructured and revised. The revised enumeration scheme bears several benefits: (1) ease of description, (2) ease of implementation, (3) ease of generalization.

We proceed as follows: In Subsection 3.1 the precedence tree which guides the search for an optimal solution is presented. In Subsection 3.2, the detailed realization of the algorithm for minimizing the project's makespan is outlined and subsequently, generalized for dealing with any regular measure of performance. Moreover, we briefly summarize adaptations for dealing with generalized problem settings.

3.1 The Precedence Tree

The main problem to solve in development stage of an exact solution procedure of the branch-and-bound type is the construction of the enumeration tree. Roughly speaking, its responsibility is to guide the search for an optimal solution by the successive decomposition of the problem into subproblems by splitting the feasible region and fixing variables. Obviously, the inclusion of parts of the constraints bears benefits, even if the subproblems are dealt with standard relaxations, e.g. LP relaxation or Lagrange relaxation. The problem at hand allows a design of the control process which explicitly takes the precedence constraints into account.

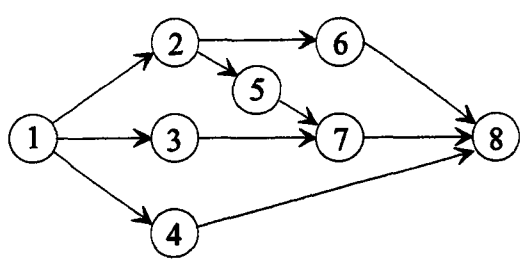

Figure 1: Example Network

For ease of notation we focus on the single-mode case and consider the example given in Figure 1 (cf. [9], p. 179). Obviously activities 2, 3 and 4 cannot be started before activity 1 is finished. If activity 1 is scheduled activity 2, 3 and 4 become eligible. In contrast to e.g. [6], [25], an activity is

called eligible if all its predecessors are scheduled but not necessarily finished. Using graph theoretical terminology the activities 2, 3 and 4 will be denoted as descendents (sons) of activity 1 (the father). These relationships are depicted by the precendence tree given in Figure 2 (cf. [15], p. 12).
On each level exactly one activity out of the set of eligible activities is scheduled. On the first level only activity 1 is eligible and on the second (after activity 1 is scheduled) the activities 2, 3 and 4 become eligible. If we now schedule activity 2, then additionally activities 5 and 6 become eligible. If we schedule activity 3 or 4 on the second level, then no additional activities become eligible. E.g. activity 5 and 6 are not eligible because not all of their predecessors (activity 2) are scheduled.

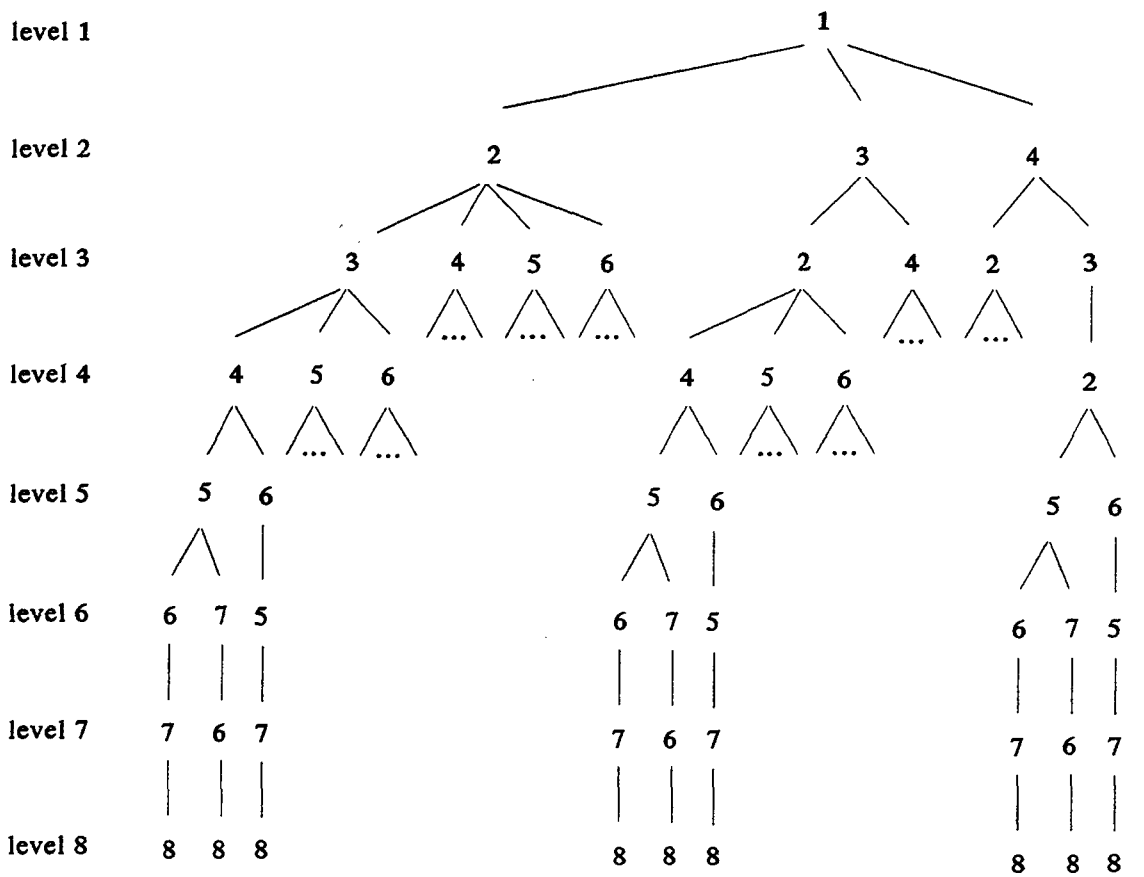

Figure 2: Precedence Tree

Since we have exactly one start activity, we can successively determine the eligible set (set of eligible activities) Y_i of level i, $i = 1, \ldots, J$. Of course, the eligible sets Y_i depend on the "history", i.e. the set of already scheduled activities.
We denote with g_i the number of the activity scheduled on level i and the set of activities currently scheduled up to level i is abbreviated to $\mathcal{ACS}_i$. Note, since we have assumed that the network is

numerically labelled, activity 1 is always the unique activity eligible on level 1.

$$
\begin{aligned}
Y_1 &:= \{1\} \\
\mathcal{ACS}_1 &:= \{g_1\} = \{1\} \\
Y_{i+1} &:= Y_i \backslash \{g_i\} \cup \{k \in S_{g_i}; \mathcal{P}_k \subset \mathcal{ACS}_i\} && \mathrm{i} = 1, \ldots, \mathrm{J}-1 \\
\mathcal{ACS}_{i+1} &:= \mathcal{ACS}_i \cup \{g_i\} && \mathrm{i} = 1, \ldots, \mathrm{J}-1
\end{aligned}
$$

Remark 1

The essential limit is that not all the successors of the scheduled activity g_i become eligible on level $i+1$. In the example given above, if activity 3 is scheduled on level 2 then none of its successors becomes eligible (cf. [15], p. 14).

The problem now arising is when to start activity g_i on level i. Due to the precedence constraints activity g_i can only be started after the completion of all its predecessors, but, inspite of this limitation several feasible start times can be assigned to the activity currently under consideration (cf. [24], pp. 39). Nevertheless, the investigation can be reduced to a single alternative if a regular measure of performance is considered as objective.

We extend our considerations to the multi-mode case and state the theorem, which, roughly speaking, reduces the examination of one path from the root to a leave of the precedence tree to at most one start time assignment per activity.

Theorem 1

Let (P) be a scheduling problem of the type GRCPSP. If (P) is feasible, then there exists a permutation of the activities $1, \ldots, J$, denoted as $g_1, \ldots, g_J$, and accompanying modes $m_{g_1}, \ldots, m_{g_J}$, such that the schedule with start times $ST_{g_1}, \ldots, ST_{g_J}$ fulfilling

(a) $ST_{g_i} \leq ST_{g_{i+1}}$, $\qquad i = 1, \ldots, J-1$,

(b) ST_{g_i}, $i = 1, \ldots, J$, *is the* **lowest** *feasible start time of activity g_i scheduled in mode m_{g_i} with respect to the precedence relations and the leftover capacities after scheduling activities $g_1, \ldots, g_{i-1}$ and (a)*

represents an optimal solution.

Proof: Since the proof is more technical we will only sketch it out. Due to the feasibility of the problem we can choose an optimal solution with start times $ST_1^*, \ldots, ST_J^*$ and modes $m_1^*, \ldots, m_J^*$. We then order the jobs with respect to nondecreasing start times and define the corresponding permutation. If the permutation fulfills the requirements, that is (b), then we are finished. Otherwise we choose

the lowest index i of g_i, $i = 1,\ldots,J$, which vioalates (b). Let k denote this index. Due to the constant per-period request we can shift activity g_k to the left, i.e. we can start it earlier, as far as (a) holds, and the newly derived start time fulfills (b) for $j = 1,\ldots,k$. Since the objective function is a regular measure of performance the objective function value does not increase. Repeatedly applying the procedure leads to an optimal solution which fulfills the requirements. □

The search space reduction by Theorem 1 is substantial (cf. [24], pp. 39). However, note, the start times (of the activities) in conditions (a) and (b) of Theorem 1 cannot be replaced by their completion times as suggested in [15] without loosing optimality (cf. [24], pp. 37).

3.2 The Enumeration Scheme

Using the preliminaries presented in the previous subsection we can concisely state the algorithm: The algorithm schedules one activity per node of the branch-and-bound tree. An activity is firstly considered for scheduling in its first mode when all of its predecessors are scheduled. The start time of the activity under consideration is the lowest feasible start time which (a) is not less than the start time of the activity most recently scheduled and (b) does not violate the precedence or resource constraints. If scheduling in the current mode is infeasible then the next mode is tested. If there is no untested mode left then the next eligible activity is selected. If there is no untested eligible activity left then backtracking to the previous level is performed. At this level the next mode or eligible activity is chosen.

The summary of the algorithm turns out a degree of freedom concerning the selection of the activity, more precisely the activity/mode combination, to be considered for scheduling on the current level first, second and so forth. Using priority values (cf. e.g. [3], [24], pp. 51) we can define a sequence on the job/mode combinations $[j,m], j \in Y_i, m = 1,\ldots,M_j$. Note, in the description given above the modes of an activity are examined consecutively. Avoiding notational overhead we are leaving priority rules for later discussion and only refer to if there is the necessity for doing so.

We will now give a more formal description which will enable us to elaborate the kernel of the enumeration scheme. Especially the representation of the bounding rules to be presented in Section 4 will be substantially simplified. The notation used to describe the algorithm is displayed in Table 3.

The algorithm for the minimization of the makespan is presented in Table 4. In Step 1 we initialize the variables used. We avoid the case distinction whether any activity is already scheduled or not by defining $g_0 := 0$, $ST_{g_0} := 0$, $CT_{g_0} := 0$ and $\mathcal{P}_1 := \{0\}$. The set of eligible activities on the first level ($i = 1$) is defined as $Y_1 := \{1\}$ with corresponding number of eligible activities within the first level $\hat{N}_1 := |Y_1| = 1$. The first and only activity to be tested is the unique start activity, thus we

i	: level index
i^*	: lowest index which produces a time window violation after the recalculation
Y_i	: set of eligible activities on level i
$\hat{N}_i$	: cardinality of the set Y_i
N_i	: index of the element from the eligible set Y_i which is currently under consideration
Y_{iN_i}	: the N_i'th element of the set of eligible activities on level i
$\mathcal{ACS}$ ($\mathcal{ACS}_i$)	: set of activities currently scheduled (up to level i)
g_i	: activity currently scheduled or under consideration on level i
m_{g_i}	: number of the mode activity g_i is currently scheduled in or considered to be scheduled in on level i
ST_{g_i} (CT_{g_i})	: start (completion) time of activity g_i scheduled on level i
t_P	: lowest feasible start time of activity g_i with respect to the precedence relation
t_I	: lowest feasible start time of activity g_i with respect to condition (a) of Theorem 1
Φ^*	: objective function value of the currently best known solution
$\overline{\Phi}(\cdot)$	: function, which assigns each partial schedule a lower bound on the objective function value of a completion; the bound is equal to the objective function value, if a complete schedule is considered.

Table 3: Notation Used in the B&B-Algorithm

have $g_1 = Y_{11} = 1$. In general, we denote with Y_{iN_i} the N_i'th element of the eligible set Y_i where the activities in Y_i, except where stated otherwise, are arranged with respect to increasing job number. In Step 2 a job/mode combination is selected for assignment. Two cases have to be distinguished. First, if there are further modes for the activity (descendant) currently examined, that is, $m_{g_i} < M_{g_i}$, then the next mode is selected, i.e. $m_{g_i} := m_{g_i} + 1$. Second, if for the current descendant g_i the last mode has been tested, that is, $m_{g_i} = M_{g_i}$, then the next descendant is chosen and the first mode is selected for assignment, i.e. $N_i := N_i + 1$, $g_i := Y_{iN_i}$, and $m_{g_i} := 1$. If no more untested modes and descendants are available, then one-level backtracking (Step 3) has to be performed, otherwise we goto Step 4.

In Step 4 we first calculate the lowest feasible start time t_P with respect to the currently scheduled activities and the precedence relations, that is, $t_P := max\{CT_k; k \in \mathcal{P}_{g_i}\}$. Subsequently we calculate the minimal feasible start time t_I due to condition (a) of Theorem 1 and define $t^* := max\{t_P, t_I\}$. We then scan the interval $[t^*, LF_{g_i} - d_{g_i m_{g_i}}]$ for the earliest contiguous interval $d_{g_i m_{g_i}}$ periods long where activity g_i can be scheduled in mode m_{g_i} without violating the renewable resource constraints. That is, we search $\bar{t}$, $t^* \leq \bar{t} \leq LF_{g_i} - d_{g_i m_{g_i}}$, such that the leftover capacities of the renewable resources r, $r \in R$,

Step 1: (Initialization)

$\mathcal{ACS} := \emptyset$; $g_0 := 0$; $ST_{g_0} := 0$; $CT_{g_0} := 0$; $\mathcal{P}_1 := \{0\}$; $i := 1$; $Y_1 := \{1\}$; $\hat{N}_1 := 1$; $N_1 := 1$; $g_1 := 1$; $m_1 := 0$;

Step 2: (Select next untested mode or descendant)

If $m_{g_i} < M_{g_i}$ then $m_{g_i} := m_{g_i} + 1$ and goto Step 4;

If $N_i < \hat{N}_i$ then $N_i := N_i + 1$; $g_i := Y_{iN_i}$; $m_{g_i} := 1$ and goto Step 4;

Step 3: (One-level backtracking)

$i := i - 1$; if $i = 0$ then STOP, else remove job g_i from partial schedule; $\mathcal{ACS} := \mathcal{ACS} \backslash \{g_i\}$; readjust resource arrays and goto Step 2;

Step 4: (Find feasible start time)

$t_P := \max\{CT_k; k \in \mathcal{P}_{g_i}\}$; $t_I := ST_{g_{i-1}}$; $t^* := \max\{t_P, t_I\}$; determine the **earliest** resource feasible start time $\bar{t}$, $t^* \leq \bar{t} \leq LF_{g_i} - d_{g_i m_{g_i}}$, of job g_i in mode m_{g_i}; if scheduling is impossible goto Step 2, else set $ST_{g_i} := \bar{t}$; $CT_{g_i} := \bar{t} + d_{g_i m_{g_i}}$; $\mathcal{ACS} := \mathcal{ACS} \cup \{g_i\}$ and adjust resource arrays;

Step 5: If i=J then goto Step 7;

Step 6: (Update the eligible set)

$i := i + 1$; calculate the new descendant set $Y_i := Y_{i-1} \backslash \{g_{i-1}\} \cup \{k \in \mathcal{S}_{g_{i-1}}; \mathcal{P}_k \subseteq \mathcal{ACS}\}$; set $\hat{N}_i := |Y_i|$; $N_i := 1$; $g_i := Y_{i1}$; $m_{g_i} := 0$ and goto Step 2;

Step 7: (Store solution and adjust time bounds)

Store solution g_j, m_{g_j}, ST_{g_j}, $j = 1, \ldots, J$;

Set $LS_j := LS_j - (LF_J - CT_J + 1)$, $j = 1, \ldots, J$;

and $LF_j := LF_j - (LF_J - CT_J + 1)$, $j = 1, \ldots, J$;

Step 8: (Calculate lowest indexed level violating the time window)

$i^* := \min\{k \in \{1, \ldots, J\}; CT_{g_k} > LF_{g_k}\}$;

Step 9: (Variable-level backtracking)

Readjust resources used (consumed) by jobs g_k in mode m_{g_k}, $k = J, \ldots, i^*$;

$\mathcal{ACS} := \mathcal{ACS} \backslash \{g_J, \ldots, g_{i^*}\}$; $i := i^*$ and goto Step 2.

Table 4: Minimizing the Project's Makespan

in the periods t, $t = \bar{t}+1, \ldots, \bar{t}+d_{g_i m_{g_i}}$, are at least equal to the per-period requirements $k^{\rho}_{g_i m_{g_i} r}$. Note, as previously mentioned, since different modes may have different durations, searching the interval $[t^*, LS_{g_i}]$ for a start time is not equivalent to searching the interval $[t^*, LF_{g_i} - d_{g_i m_{g_i}}]$. In the former case it might happen that the bounds exposed by the latest finish times are violated without detecting it on

the current level, and thus producing computational overhead. Additionally, feasibility with respect to nonrenewable resources has to be checked, i.e. the leftover capacities of the nonrenewable resources r, $r \in N$, are compared with the consumption $k^{\nu}_{g_i m_{g_i} r}$ of the activity currently considered. Obviously, since determining a feasible start time with respect to renewable resources usually consumes more time than comparing consumptions and the remaining availabilities of the nonrenewable resources, it is sensible to check the latter ones first.

If feasibility can be assured, we set the start time ST_{g_i} of activity g_i equal to $\bar{t}$ and the completion time CT_{g_i} equal to $ST_{g_i} + d_{g_i m_{g_i}}$. Furthermore, the set of activities currently scheduled $\mathcal{ACS}$, is updated, i.e. $\mathcal{ACS} := \mathcal{ACS} \cup \{g_i\}$, and the leftover capacities are adjusted. If feasibility cannot be assured, we return to Step 2 and determine the next job/mode combination. Successfully scheduling of activity g_i leads to Step 5.

In Step 5 we check if all the activities are scheduled. If this holds true, we have found the first feasible solution or an improved solution and we can skip to Step 7, where the new solution is stored and the critical path bounds LS_j, LF_j, $j = 1, \ldots, J$, are recalculated. Since our goal is the improvement of the currently best known solution LS_j and LF_j are reduced by the improvement of the former best known solution incremented by one, that is $LF_J - CT_J + 1$.

If not all the activities are scheduled, we step over to the next level, $i := i + 1$, and calculate the new descendants, i.e. set of eligible activities Y_i, the number of descendants $\hat{N}_i := |Y_i|$, select the first descendant $N_i := 1$, $g_i := Y_{i1}$, initialize the mode selector variable $m_{g_i} := 0$ and goto Step 2.

After the adaptation of the critical path bounds in Step 7 we determine the lowest level index i^*, where the newly derived critical path bounds are violated by the current schedule. Variable-level backtracking is then performed in Step 9, that is, all the activities g_i, which have been scheduled on a level i, $i \geq i^*$, are removed from the schedule, the resource availabilities are readjusted and the current level index is set equal to i^*.

The algorithm terminates if in Step 2 no more job/mode assignment is possible and decrementing the level index in Step 3 leads to $i = 0$. Otherwise, if the level index is greater than zero after decrementing it in Step 3, then g_i is removed from the partial schedule and resource availabilities are readjusted.

A thorough study of the enumeration scheme shows that its current state is entirely described by (a) the currently best known solution and bound on the objective function, (b) the sequences of the job/mode combinations the eligible activities have to be examined in, and (c) the job/mode combinations $[g_j, m_{g_j}]$, $j = 1, \ldots, i$, scheduled up to the current level i. The latter information and its implications are summarized in the following definitions.

Definition 2 *(cf. [24], pp. 42)*

(a) An i-partial schedule $\mathcal{PS}_i$ is an $4 \times i$-matrix with columns $(j, g_j, m_{g_j}, ST_{g_j})$, $j = 1, \ldots, i$, defining that job g_j is scheduled on level j in mode m_{g_j} with start time ST_{g_j}.

(b) An i-partial schedule $\mathcal{PS}_i$, where all the activities are scheduled, i.e. $i = J$, is called schedule.

(c) Let $\mathcal{PS}_i$ be an i-partial schedule. A j-partial schedule $\mathcal{PS}_j$, $j \geq i$, with first i columns coinciding with the ones of $\mathcal{PS}_i$ is called a continuation of the i-partial schedule $\mathcal{PS}_i$. If $j = J$ then the continuation is called a completion of $\mathcal{PS}_i$.

(d) A completion $\mathcal{PS}_J$ of an i-partial schedule $\mathcal{PS}_i$ is called optimal completion if no completion of $\mathcal{PS}_i$ has an objective function value better than the one of $\mathcal{PS}_J$.

Definition 3

Let $\overline{T}$ denote an upper bound on the project's makespan and $\mathcal{PS}_i$ be an i-partial schedule with corresponding set of currently scheduled activities $\mathcal{ACS}(\mathcal{PS}_i) := \bigcup_{j=1}^{i} \{g_j\}$.

(a) $\mathcal{PS}_i$ is called precedence feasible, if the precedence constraints are met, that is, $ST_{g_h} + d_{g_h m_{g_h}} \leq ST_{g_k}$, $g_h, g_k \in \mathcal{ACS}(\mathcal{PS}_i)$ and $g_h \in \mathcal{P}_{g_k}$.

(b) $\mathcal{PS}_i$ is called feasible with respect to the nonrenewable resources, if the cumulated consumption of the currently scheduled job/mode combinations does not exceed the availability of any nonrenewable resource. The remaining capacity $K_r^{\nu}(\mathcal{PS}_i) = K_r^{\nu} - \sum_{j=1}^{i} k_{g_j m_{g_j} r}^{\nu}$ is called leftover capacity of the nonrenewable resource r, $r \in N$, with respect to i-partial schedule $\mathcal{PS}_i$.

(c) $\mathcal{PS}_i$ is called feasible with respect to the renewable resources, if the cumulated usage of the currently scheduled job/mode combinations does not exceed the availabilty of any resource in any period. The remaining capacity $K_{rt}^{\rho}(\mathcal{PS}_i) = K_{rt}^{\rho} - \sum_{\substack{j=1 \\ ST_{g_j}+1 \leq t \leq ST_{g_j}+d_{g_j m_{g_j}}}}^{i} k_{g_j m_{g_j} r}^{\rho}$ is called leftover capacity of the renewable resource r, $r \in R$, in period t, $t = 1, \ldots, \overline{T}$, with respect to i-partial schedule $\mathcal{PS}_i$.

(d) $\mathcal{PS}_i$ is called feasible, if it is precedence feasible and feasible with respect to the nonrenewable and the renewable resources.

Clearly, in accordance with Theorem 1 and Step 4 of the algorithm it is not necessary to keep the start time of the activities. However, the enhanced representation will simplify the later discussion and the implementation of acceleration schemes.

Using the definitions we can describe the construction of partial schedules as follows: On each level $(i+1)$ of the branch-and-bound tree a feasible i-partial schedule $\mathcal{PS}_i$ is extended to a feasible $(i+1)$-partial schedule by scheduling an eligible activity g_{i+1}, $g_{i+1} \in Y_{i+1}$, in a mode $m_{g_{i+1}}$, $1 \leq m_{g_{i+1}} \leq$

$M_{g_{i+1}}$. In accordance with Theorem 1, conditions (a) and (b), the start time $ST_{g_{i+1}}$ obtained in Step 4 is uniquely determined. Thus we can denote the continuation of an i-partial schedule $\mathcal{PS}_i$ to an $(i+1)$-partial schedule $\mathcal{PS}_{i+1}$ by scheduling a job/mode combination $[g_{i+1}, m_{g_{i+1}}]$ at the earliest precedence and resource feasible start time $ST_{g_{i+1}}$ fulfilling $ST_{g_{i+1}} \geq ST_{g_i}$ by using the operator $\oplus$:

$$\mathcal{PS}_{i+1} \quad := \quad \mathcal{PS}_i \oplus [g_{i+1}, m_{g_{i+1}}]$$

We will adapt the algorithm presented for optimizing any regular measure of performance. Due to Theorem 1 only minor changes are necessary in order to attack scheduling problems of the type GRCPSP. That is, the enumeration is again guided by the precedence tree and only the evaluation of bounds on the objective function value has to be adapted and explicitly incorporated (Step 4 and Steps 7-9), respectively.

Performing Step 4 as described above follows two intensions. On the one hand, if an activity g_i (in mode m_{g_i}) is assigned a start time $\bar{t}$, $\bar{t} \leq LF_{g_i} - d_{g_i m_{g_i}}$, then the partial schedule $\mathcal{PS}_i$ is at least completable with respect to the precedence relations. On the other hand, since the latest finish times are adjusted after the improvement of the current best solution, the adapted latest start and finish times (Step 7) offer a new bound on the objective function value (makespan) obtainable from a completion of the current partial schedule. That is, if we would assign a start time $\bar{t}$, $\bar{t} > LF_{g_i} - d_{g_i m_{g_i}}$, to an activity g_i then the partial schedule is not completable with a makespan T, $T \leq LF_J$.

For notational convenience, we denote the objective function with Φ and let Φ^* denote the objective function value of the current best solution. Furthermore, $\overline{\Phi}(\cdot)$ denotes a function which determines a lower bound on the objective function of the completion for a given partial schedule $\mathcal{PS}_i$. The bound matches with the objective function value of $\mathcal{PS}_i$ if a complete schedule is the argument, i.e. $i = J$. With this in mind, we rearrange Step 4 and Steps 7-9 to Step 4' and Step 7' (cf. Table 5), respectively.

Remark 2

In contrast to [15], pp. 13, we included lower bound evaluation explicitly and prevent computational overhead by restarting the procedure after determining an improved best solution.

Although, the (modified) enumeration scheme can deal with a wide variety of scheduling problems, further modifications are necessary if e.g. (maximum) time-lags between the activities or time varying resource requests for renewable resources have to be taken into acoount (cf. [24], pp. 67). Both extensions of the model require beside the job/mode combination $[g_i, m_{g_i}]$ currently considered a third dimension which saves the start time ST_{g_i} presently evaluated. That is, before switching to another

Step 4': (Find feasible start time)

$t_P := \max\{CT_k; k \in \mathcal{P}_{g_i}\}$; $t_I := ST_{g_{i-1}}$; $t^* := \max\{t_P, t_I\}$; determine the **earliest** resource feasible start time $\bar{t}$, $t^* \leq \bar{t} \leq LF_{g_i} - d_{g_i m_{g_i}}$, of job g_i in mode m_{g_i}; if scheduling is impossible or $\overline{\Phi}(\mathcal{PS}_i) \geq \Phi^*$ then goto Step 2, else set $ST_{g_i} := \bar{t}$; $CT_{g_i} := \bar{t} + d_{g_i m_{g_i}}$; $\mathcal{ACS} := \mathcal{ACS} \cup \{g_i\}$ and adjust resource arrays;

Step 7': (Store solution and adjust the bounds)

Store solution g_j, m_{g_j}, ST_{g_j}, $j = 1, \ldots, J$, Φ^*; $i = J - 1$;

remove job g_J from partial schedule; $\mathcal{ACS} := \mathcal{ACS} \backslash \{g_J\}$; readjust resource arrays and goto Step 2;

Table 5: Optimizing a Regular Measure of Performance

job/mode combination the remaining feasible start times have to be examined. Another job/mode combination for the current job/mode combination is chosen if no further feasible start times are left. However, doing so, only minor changes are necessary to optimize any objective and taking into account time-varying requests as well as time-lags. In contrast, job specific constraints on the start and finish times imposed by a release date and a deadline can be implemented by simple time window adaptations.

4 Search Tree Reduction

By the exclusion of partial schedules from further continuation we can reduce the enumeration tree guiding the search for an optimal solution. Clearly, for optimizing the given objective we have to assure that the reduction does not make worse the (optimal) solutions obtainable. That is, we have to elaborate conditions on which a partial schedule need not be extended without loosing optimality. These conditions build the bounding rules we will present, they will be stated as theorems. Clearly, using the bounding rules is not beneficial if checking the assumptions requires more time than the truncation of the branches saves. Therefore, if necessary, additional hints for the implementation of the rules will be given.

In this section, if not otherwise mentioned, we consider regular measures of performance. Moreover, we use the notion of an i-partial schedule to refer to a feasible i-partial schedule derived by the algorithm presented in Tables 4 and 5, respectively. That is, the start time assigned to an activity is the lowest start time fulfilling conditions (a) and (b) of Theorem 1.

We distinguish two types of bounding rules, the static and the dynamic ones. Static rules can be implemented by the adaptation of the input data. The enumeration scheme remains unchanged and the rules can be employed by any algorithm solving the problem at hand. In contrast, the dynamic rules are incorporated into the algorithm. As a rule their sensible application depends on the algorithm employed.

4.1 Static Search Tree Reduction Techniques

The first static rule excludes modes and/or nonrenewable resources from the input data. That is, the number of (partial) schedules and/or constraints to be considered are reduced before the solution procedure is started. Although the rule is quite simple it might help increasing the algorithm's performance, especially when a large project with a number of activities, hardly to manage manually, is dealt with. Moreover, we believe, that working with the related definitions will bring out more useful conditions to reduce the computational time needed. We define:

Definition 4 (cf. [25]) *Let m_j, $1 \leq m_j \leq M_j$, be a mode of an activity j, $1 \leq j \leq J$.*

(a) Mode m_j is called non-executable if in a feasible schedule activity j cannot be performed in mode m_j.

(b) Mode m_j is called inefficient with respect to objective Φ, if there is another mode m'_j, $1 \leq m'_j \leq M_j$, of activity j, such that any feasible schedule $\mathcal{PS}_J$ with activity j being performed in mode m_j has a corresponding feasible schedule $\overline{\mathcal{PS}}_J$ with activity j being performed in mode m'_j and $\Phi(\overline{\mathcal{PS}}_J) \leq \Phi(\mathcal{PS}_J)$.

(c) A resource r, $r \in N \cup R$, is called redundant, if the constraints related to resource r can be neglected without influence on the set of feasible schedules.

As a consequence of Definition 4 we can now offer criteria identifying non-executable or inefficient modes and redundant resources:

Lemma 1 *Let m_j, $1 \leq m_j \leq M_j$, be a mode of activity j, $1 \leq j \leq J$.*

(a) Mode m_j is non-executable w.r.t. renewable resource r, $r \in R$, if for each start time ST_j, $ES_j \leq ST_j \leq LF_j - d_{jm}$, there is a period t, $ST_j + 1 \leq t \leq ST_j + d_{jm}$, with $k^{\rho}_{jm_jr} > K^{\rho}_{rt}$.

(b) Mode m_j is non-executable w.r.t. nonrenewable resource r, $r \in N$, if

$$k^{\nu}_{jm_jr} + \sum_{\substack{i=1 \\ i \neq j}}^{J} \min_{m=1}^{M_i} \{k^{\nu}_{imr}\} > K^{\nu}_{r}.$$

(c) Mode m_j is inefficient with respect to objectives (a) through (d) given in Section 2, if there is another mode m'_j of activity j with duration and requests at most equal to the ones of mode m_j.

Step 1:	Remove all non-executable modes from the project data.
Step 2:	Delete all redundant nonrenewable resources.
Step 3:	Eliminate all inefficient modes.
Step 4:	If any mode has been erased within Step 3, go to Step 2.

Table 6: Implementation of Theorem 2

(d) A nonrenewable resource r, $r \in N$, is redundant if the sum of the maximum consumptions of the activities, i.e. $\sum_{j=1}^{J} \max_{m=1}^{M_j}\{k_{jmr}^{\nu}\}$, does not exceed the availability K_r^{ν}.

Although a similar criterion for the exclusion of redundant renewable resources can be easily formulated we have omitted its lenghty statement and concentrated on the exclusion of nonrenewable resources. For a clear reference point when reporting our computational experience we summarize the statements in the following thereom:

Theorem 2 *(Bounding Rule 1, Input Data Reduction)*

(a) Redundant nonrenewable resources and non-executable modes can be eliminated without effect on the set of feasible solutions.

(b) Inefficient modes can be deleted without effect on the optimal objective function value obtainable.

That is, for any objective under consideration redundant resources and non-executable modes can be eliminated without changing the set of feasible schedules. In contrast, removing modes inefficient with respect to objective Φ may reduce the set of feasible schedules but does not effect the value of the optimal solution.

However, the example given in [25] shows that deleting a mode which is non-executable w. r. t. a renewable resource may force a mode of another activity to become non-executable w. r. t. a nonrenewable resource. Moreover, removing a non-executable mode from the project data may cause redundancy of a nonrenewable resource. Finally, erasing a redundant nonrenewable resource may lead to inefficiency of a mode, while eliminating an inefficient mode may cause redundancy of a nonrenewable resource. Therefore, the project's input data should be prepared by the steps described in Table 6 (cf. [25]).

The next bounding rule to be presented is designed for instances with nonrenewable resources, that is, $|N| > 0$. It checks completability of the partial schedule currently considered. The dynamic variant, as given in Theorem 3, has been proposed by Drexl (cf. [7]) for a less general model. Sprecher (cf. [24])

adapted the rule to the GRCPSP and substantially increased its efficiency by reformulating it as a preprocessing, i.e. static, rule.

Theorem 3 *(Bounding Rule 2, Input Data Adjustment)*
Let $\mathcal{PS}_i$ be a feasible i-partial schedule with left-over capacities of nonrenewable resources $K_r^\nu(\mathcal{PS}_i)$, $r \in N$. Let $\overline{ACS} = \{1,\ldots,J\}\backslash\{g_1,\ldots,g_i\}$ be the set of currently unscheduled activities. If there is a resource r, $r \in N$, with

$$K_r^\nu(\mathcal{PS}_i) \;<\; \sum_{j\in\overline{ACS}} \min\{k_{jmr}^\nu; m = 1,\ldots,M_j\}$$

then the schedule is not completable.

Since the rule checks completability of the current partial schedule it is applicable to any (regular) objective. Including the following remark the dynamic formulation can be transferred to a static one.

Remark 3
The bounding rule of Theorem 3 can be easily implemented via preprocessing by calculating:

$$kmin_{jr}^\nu := \min\{k_{jmr}^\nu; m = 1,\ldots,M_j\}, \qquad j = 1,\ldots,J, r \in N,$$

and adjusting the input data as follows:

$$\overline{k}_{jmr}^\nu := k_{jmr}^\nu - kmin_{jr}^\nu, \qquad j = 1,\ldots,J, m = 1,\ldots,M_j, r \in N$$

and

$$\overline{K}_r^\nu := K_r^\nu - \sum_{j=1}^{J} kmin_{jr}^\nu, \qquad r \in N.$$

Applying Theorem 3 and Remark 3 causes that for each pair of a nonrenewable resource r, $r \in N$, and an activity j, $j = 1,\ldots,J$, there is at least one mode m with an adapted consumption of zero units, that is, the resource factor (cf. Part II, Section 2) of the instance is reduced.
Obviously, the effect of commonly employing Bounding Rules 1 and 2 is the strongest, if, first, the input data is reduced in accordance with Theorem 2 and, second, then adjusted with respect to Theorem 3.

4.2 Dynamic Search Tree Reduction Techniques

Now we will present the dynamic search tree reduction schemes. For illustrational purpose, we enhance the instance given in Figure 1 by the data given in Table 7 and consider the related makespan

	J	$\overline{T}$		$\|R\|$		K_1^ρ		K_2^ρ		$\|N\|$			
	8	35		2		2		3		0			
Job	1	2		3		4		5		6		7	8
LF_j	19	23		27		35		27		35		35	35
Mode	1	1	2	1	2	1	2	1	2	1	2	1	1
Duration	0	4	4	6	6	5	6	4	7	3	4	8	0
k_{jm1}^ρ	0	0	1	0	1	2	1	1	1	1	0	1	0
k_{jm2}^ρ	0	3	0	2	0	1	1	2	1	2	2	3	0

Table 7: Problem Instance

minimization problem. The eligible sets are assumed to be ordered with respect to increasing job number.

Since the availabilities of the renewable resources are constant, we obtain an upper bound on the project's optimal makespan $\overline{T}$ by adding the maximal durations of the activities, that is, $\overline{T} = 35$. Note, in general, if only renewable resources have to be taken into account, the bound can be improved to the sum of the minimum activity durations related to executable modes, that is, $\overline{T} = 30$. However, using the first branch $[1,1]$, $[2,1]$, $[3,1]$, $[4,1]$, $[5,1]$, $[6,1]$, $[7,1]$ and $[8,1]$ we determine the start times $ST_1 = 0$, $ST_2 = 0$, $ST_3 = 4$, $ST_4 = 4$, $ST_5 = 10$, $ST_6 = 14$, $ST_7 = 17$ and $ST_8 = 25$. The first feasible solution is found. It is displayed in Figure 3, where the brackets in the Gannt-chart denote $[g_i, m_{g_i} | k^\rho_{g_i m_{g_i} 1}, k^\rho_{g_i m_{g_i} 2}]$.

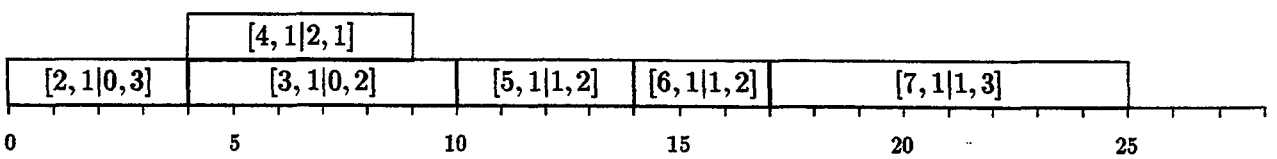

Figure 3: Feasible Solution 1

Using the new bound on the makespan, we can recalculate the time windows in accordance with Step 7 of the algorithm given in Table 4. We obtain $LF_1 = 8$, $LF_2 = 12$, $LF_3 = 16$, $LF_4 = 24$, $LF_5 = 16$, $LF_6 = 24$, $LF_7 = 24$, and $LF_8 = 24$.

Proceeding the enumeration process the second solution, displayed in Figure 4, with a makespan of 24 periods is determined. The latest finish times are adapted to $LF_1 = 7$, $LF_2 = 11$, $LF_3 = 15$, $LF_4 = 23$, $LF_5 = 15$, $LF_6 = 23$, $LF_7 = 23$, and $LF_8 = 23$.

We continue the enumeration and examine the path of the precedence tree determined by successively

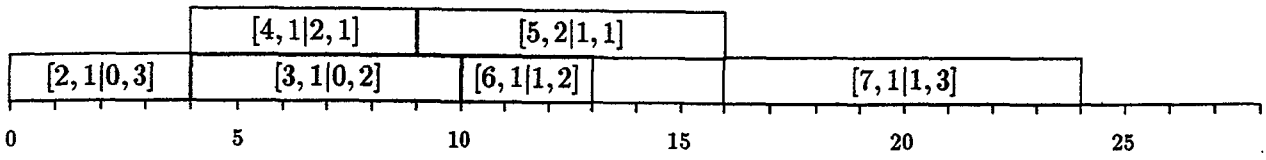

Figure 4: Feasible Solution 2

scheduling [1,1], [2,1], [3,1], [5,1], [7,1], and [4,1] or [4,2]. The related schedules are displayed in Figure 5. Since [4,1] and [4,2] cannot be scheduled without violating the resource and time window

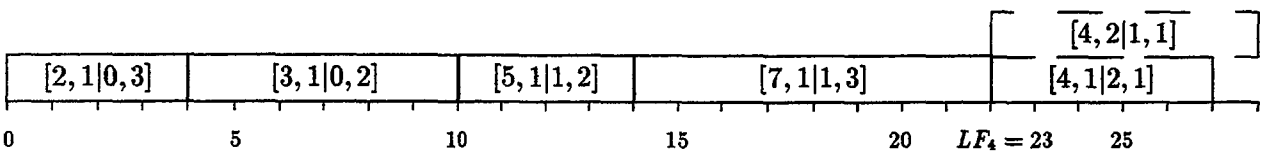

Figure 5: Non-Delayability Rule

constraints, i.e. the bound imposed by $LF_4 = 23$, activity 4 cannot be delayed to a higher level by scheduling activity 6 on the current level ($i = 6$). Intermediately scheduling of an activity/mode combination on level 6 would tighten the time bounds activity 4 can be scheduled in and, furthermore, would reduce the resources available for the execution of activity 4. Thus after recognizing that an activity is not schedulable backtracking should occur in order to free resources for its completion. That is, an activity that cannot be scheduled on the current level is not delayable for scheduling on a higher one. The result is summarized in the following Theorem 4.

Theorem 4 *(Bounding Rule 3, Non-Delayability Rule)*

If a job g_{i+1} is not schedulable (w.r.t. precedence-, time window- or resource-constraints) on level $(i+1)$ with feasible i-partial schedule $\mathcal{PS}_i$ then it is not schedulable on level $(i+k+1)$ with $(i+k)$-partial schedule $\mathcal{PS}_{i+k}$ where the first i columns of $\mathcal{PS}_{i+k}$ match with those of $\mathcal{PS}_i$.

Proof: Obvious □

Surely, the underlying idea is adaptable to a single job/mode combination, that is, if a job g_{i+1} is not schedulable in mode $m_{g_{i+1}}$ on level $(i+1)$, with i-partial schedule $\mathcal{PS}_i$, then $[g_{i+1}, m_{g_{i+1}}]$ is not schedulable on a level $(i+k+1)$ with $(i+k)$-partial schedule $\mathcal{PS}_{i+k}$. Unfortunately, the effect of this adaptation is completely consumed by the additional effort to be performed (cf. Part II, Section 3). We carry on the enumeration and return via single level backtracking to level 4, where activity 5 is rescheduled in mode 2 as a result of which the partial schedule in Figure 6 is derived. Recall, the partial schedule is obtained by scheduling [1,1], [2,1], [3,1] and then [5,2]. However, successively scheduling [1,1], [2,1], [5,2] and [3,1] would result in the same partial schedule. That is, since the same

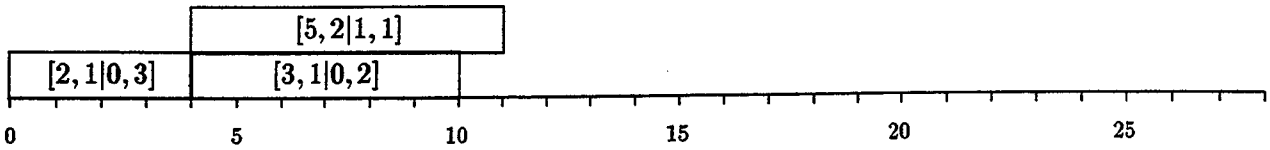

Figure 6: Single Enumeration Rule

completions are obtained from the latter sequence it should be excluded from further continuation. The fact itself is simply stated in Theorem 5 and needs no further explanation.

Theorem 5 *(Bounding Rule 4, Single Enumeration Rule)*

Let $\mathcal{PS}_{i+2}$ *and* $\overline{\mathcal{PS}}_{i+2}$, $i \geq 1$, *be two feasible* $(i+2)$*-partial schedules with*

$$\mathcal{PS}_{i+2} = \mathcal{PS}_i \oplus [g,m] \oplus [h,n] \quad , \quad \overline{\mathcal{PS}}_{i+2} = \mathcal{PS}_i \oplus [h,n] \oplus [g,m]$$

where $[g,m]$, $[h,n]$ *have start times* ST_g, ST_h *in* $\mathcal{PS}_{i+2}$ *and* $\overline{ST}_h$, $\overline{ST}_g$ *in* $\overline{\mathcal{PS}}_{i+2}$, *respectively. If* $ST_g = ST_h = \overline{ST}_h = \overline{ST}_g$ *then the completions of* $\overline{\mathcal{PS}}_{i+2}$ *are dominated by the completions of* $\mathcal{PS}_{i+2}$.

Proof: Obvious □

For a less general model, the rule has already been mentioned in the literature (cf. [7]). Obviously, it is extendable: We consider an i-partial schedule $\mathcal{PS}_i$ and unscheduled activities $g_{i+1},\ldots,g_{i+k}$, with accompanying modes $m_{g_{i+1}},\ldots,m_{g_{i+k}}$. If the start times $ST_{g_{i+1}},\ldots,ST_{g_{i+k}}$ of $g_{i+1},\ldots,g_{i+k}$, scheduled on levels $i+1,\ldots,i+k$, in modes $m_{g_{i+1}},\ldots,m_{g_{i+k}}$, do not depend on the permutation of $g_{i+1},\ldots,g_{i+k}$, i.e. $ST_{g_{i+1}} = \ldots = ST_{g_{i+k}}$, then only one permutation has to be examined, i.e. $(k!-1)$ continuations can be saved.

The crucial problem to solve is the efficient verification of the assumptions of Theorem 5. We will introduce a very efficient test. Although the concept bases on two interchangeable activities a later example will illustrate that the generalized dominance concept is realized.

We modify the algorithm's Step 4' to Step 4" (cf. Table 8). The three-dimensional integer-array $PT[i][g][m]$ is initialized with -1.

Using the modification of the algorithm we can now present a criterion for efficiently checking the assumptions of Theorem 5.

Theorem 6

Assume $d_{jm} > 0$, $j = 2,\ldots,J-1$, $m = 1,\ldots,M_j$. *Let* $\overline{\mathcal{PS}}_{i+2}$, $i \geq 1$, $\overline{\mathcal{PS}}_{i+2} = \mathcal{PS}_i \oplus [h,n] \oplus [g,m]$, *be a feasible* $(i+2)$*-partial schedule derived by the modified algorithm. Moreover, let* $\overline{ST}_h$ *and* $\overline{ST}_g$ *denote the start time of* $[h,n]$ *and* $[g,m]$ *in* $\overline{\mathcal{PS}}_{i+2}$, *respectively. Then* $\overline{ST}_h = \overline{ST}_g$, $N_{i+1} > N_{i+2}$ *and*

Step 4":	$t_P := \max\{CT_k; k \in \mathcal{P}_{g_i}\}$; $t_I := ST_{g_{i-1}}$; $t^* := \max\{t_P, t_I\}$; determine the **earliest** resource feasible start time $\bar{t}$, $t^* \leq \bar{t} \leq LF_{g_i} - d_{g_i m_{g_i}}$, of job g_i in mode m_{g_i}; if scheduling is impossible then goto Step 2;
	$PT[i+1][g_i][m_{g_i}] := \bar{t}$, if $\overline{\Phi}(\mathcal{PS}_{i+1}) \geq \Phi^*$ then goto 2, else set $ST_{g_i} := \bar{t}$; $CT_{g_i} := \bar{t} + d_{g_i m_{g_i}}$; $\mathcal{ACS} := \mathcal{ACS} \cup \{g_i\}$ and adjust resource arrays;

Table 8: Extension of the Algorithm by Bounding Rule 4

$PT[i+2][g][m] = \overline{ST}_g$ implies the assumptions of Theorem 5, that is, the completions of $\overline{\mathcal{PS}}_{i+2}$ are dominated by previously evaluated completions of $\mathcal{PS}_{i+2}$, $\mathcal{PS}_{i+2} = \mathcal{PS}_i \oplus [g,m] \oplus [h,n]$.

Proof: Since we assumed $d_{jm} > 0$, $j = 2, \ldots, J-1$, $m = 1, \ldots, M_j$, the equality $\overline{ST}_h = \overline{ST}_g$ implies $h, g \in Y_{i+1}$.

Since activity g is schedulable on level $(i+2)$ in mode m with $(i+1)$-partial schedule $\overline{\mathcal{PS}}_{i+1} = \mathcal{PS}_i \oplus [h,n]$ and start time $\overline{ST}_g$, it is also schedulable on level $(i+1)$ with i-partial schedule $\mathcal{PS}_i$.

Let $Y_{i+1} = \{l_1, \ldots, l_{\hat{N}_{i+1}}\}$ be the (ordered) eligible set of level $(i+1)$ corresponding to the i-partial schedule $\mathcal{PS}_i$ with $r = N_{i+1}$, i.e. $l_r = h$.

Since the network is assumed to be numerically labeled and, furthermore, the eligible set is ordered with respect to increasing job numbers we now have $Y_{i+2} = \{l_1, \ldots, l_{r-1}, j_r, \ldots, j_{\hat{N}_{i+2}}\}$ with $g = l_s$, i.e. $s = N_{i+2} \leq r-1 < r = N_{i+1}$. Thus $PT[i+2][g][m]$ is the start time ST_g of g scheduled in mode m on level $(i+1)$ with i-partial schedule $\mathcal{PS}_i$. Using $\overline{ST}_g = \overline{ST}_h = PT[i+2][g][m] = ST_g$ we conclude $ST_g = ST_h$ in the $(i+2)$-partial schedule $\mathcal{PS}_{i+2} = \mathcal{PS}_i \oplus [g,m] \oplus [h,n]$. Since $N_{i+1} > N_{i+2}$ the completions of $\mathcal{PS}_{i+2}$ have been previously examined and the theorem is proven. □

Remark 4

We assume the eligible set $Y_{i+1} = \{l_1, \ldots, l_{\hat{N}_{i+1}}\}$ is ordered with respect to non-decreasing priority values $\Pi(l)$, i.e. $\Pi(l_v) \leq \Pi(l_{v+1})$, $v = 1, \ldots, \hat{N}_{i+1} - 1$. If $Y_{i+2} = \{l_1, \ldots, l_{\hat{N}_{i+1}}\} \backslash \{l_r\} \cup \{k \in \mathcal{S}_{l_r}; \mathcal{P}_k \subset \mathcal{ACS}\}$ can be ordered by priority rule Π, such that $\Pi(l_v) \leq \Pi(w)$ for all $w \in \{h \in \mathcal{S}_{l_r}; \mathcal{P}_h \subset \mathcal{ACS}\}$ and $v < r$ then Theorem 6 is also applicable when using priority rule $\Pi(\cdot)$ instead of the job number rule.

Obviously, the assumptions of Remark 4 are fulfilled if the eligible activities are selected in accordance with the minimum late finish time or minimum precedence feasible start time (cf. [24], pp. 51).

We illustrate the effect of the rule if more than two jobs have the same start time by using the activity/mode combinations $[1,1]$, $[2,2]$, $[3,1]$, $[4,2]$ and disregarding the modes of the remaining activities. Since any sequence of the activities/mode combinations $[2,2]$, $[3,1]$ and $[4,2]$ produces the

same start times $ST_2 = ST_3 = ST_4 = 0$ only one sequence has to be pondered. That is, applying Bounding Rule 4 prunes the precedence tree as depicted in Figure 7, where (—) denotes the pruning of a branch. Note, using pairwise comparisons of start times the generalized concept is realized, that is, if the conditions hold only one sequence is continued.

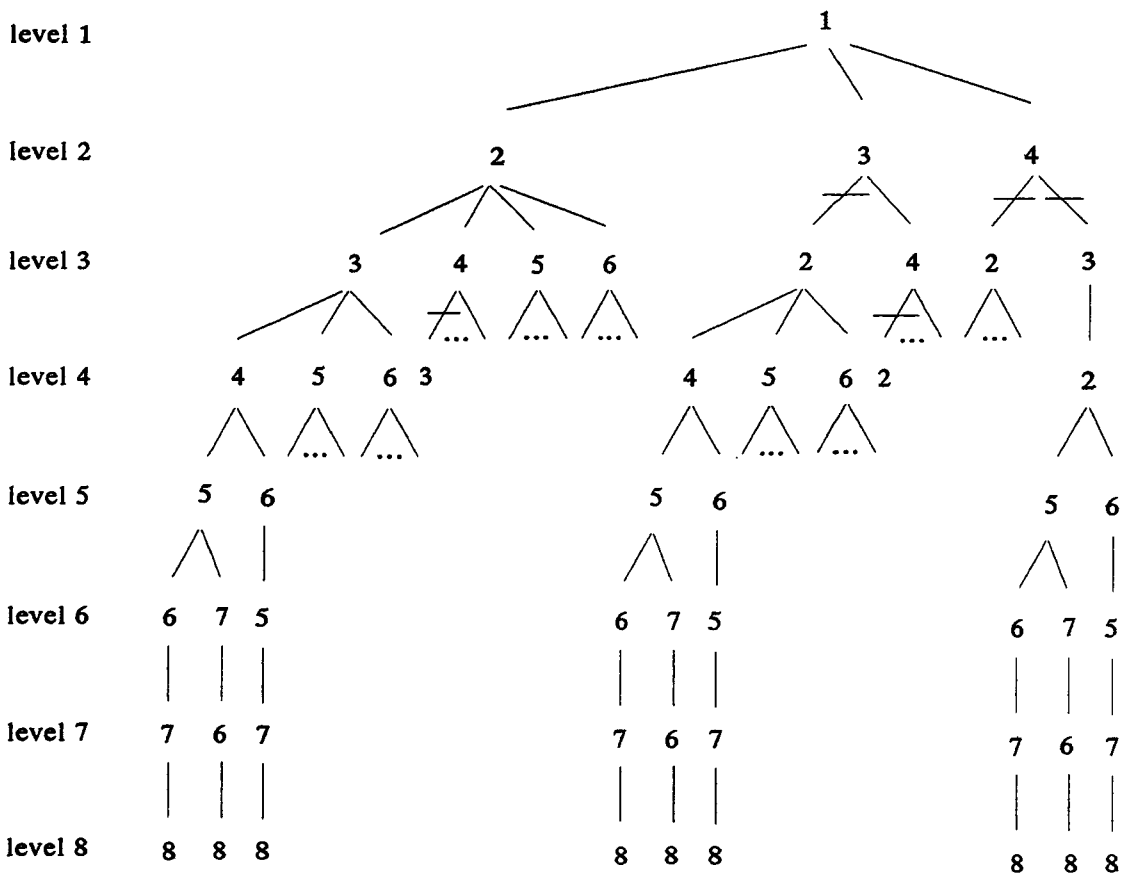

Figure 7: Precedence Tree pruned by Bounding Rule 4

The following bounding rule makes use of the fact that the set of semi-active schedules is a dominant set with respect to any regular measure of performance (cf. [26]). That is, for any regular measure of perfomance there is an optimal semi-active schedule. The bounding rule is the so called local left-shift rule.

Stepping forward with the enumeration we come to the partial schedule related to the sequence [1, 1], [2, 1], [3, 1], [5, 2], [4, 1], [6, 2], the related Gannt-chart of which is given in Figure 8. Due to Theorem 1 condition (a) it is $ST_{g_i} \leq ST_{g_{i+1}}$, i.e. $ST_6 = 11 \leq ST_{g_{i+1}}$, thus the completions derived by continuing the current partial schedule cannot be semi-active. Moreover, the left-over capacities in periods t, $t = ST_6 + 1, \ldots, \overline{T}$ are at most equal to the ones belonging to the sequence [1, 1], [2, 1], [3, 1], [5, 2],

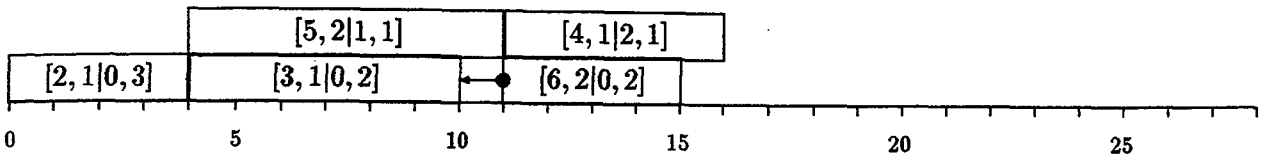

Figure 8: Local Left-Shift Rule

[6,2], [4,1].

The rule has already been mentioned and successfully implemented for the single-mode case (cf. e.g. [6], [27]). It is stated in the following theorem and the proof is given for the sake of completeness.

Theorem 7 *(Bounding Rule 5, Local Left-Shift Rule)*
Let $\mathcal{PS}_{i+1} = \mathcal{PS}_i \oplus [g_{i+1}, m_{g_{i+1}}]$ be a feasible $(i+1)$-partial schedule and let $ST_{g_{i+1}}$ denote the start time of activity g_{i+1} scheduled in mode $m_{g_{i+1}}$ in $\mathcal{PS}_{i+1}$. If, by ignoring condition (a) of Theorem 1, activity g_{i+1} is additionally schedulable in mode $m_{g_{i+1}}$ with start time $\overline{ST}_{g_{i+1}}$, $\overline{ST}_{g_{i+1}} = ST_{g_i} - 1$, without violating the precedence- and (renewable) resource-constraints, then the $(i+1)$-partial schedule $\mathcal{PS}_{i+1}$ is dominated.

Proof: Let $k := \min\{l \leq i; ST_{g_l} = ST_{g_i}\}$. Since $\mathcal{PS}_{i+1}$ is feasible the $(i+1)$-partial schedule $\overline{\mathcal{PS}}_{i+1}$, $\overline{\mathcal{PS}}_{i+1} = \mathcal{PS}_{k-1} \oplus [g_{i+1}, m_{g_{i+1}}] \oplus [g_k, m_{g_k}] \oplus \cdots \oplus [g_i, m_{g_i}]$ is a feasible (i+1)-partial schedule, too. Due to the constant per-period usage of the renewable resources we have

$$K^{\rho}_{rt}(\mathcal{PS}_{i+1}) \;\leq\; K^{\rho}_{rt}(\overline{\mathcal{PS}}_{i+1}) \qquad r \in R,\ t = ST_{g_{i+1}} + 1, \ldots, \overline{T}$$

and use $ST_{g_{i+1}} \leq ST_{g_{i+2}}$ for completing the proof. □

The local left-shift rule can be used when optimizing any regular measure of performance. It reduces the enumeration to the set of semi-active schedules (cf. [26]). In contrast, to the best of our knowledge, Demeulemeester and Herroelen's implementation (cf. [6]) might additonally generate schedules that are not semi-active.

Note, Bounding Rule 5 does not imply that the schedules $\overline{\mathcal{PS}}_{i+k+1}$ with $\overline{\mathcal{PS}}_{i+k+1} = \mathcal{PS}_i \oplus [\overline{g}_{i+1}, m_{\overline{g}_{i+1}}] \oplus \cdots \oplus [\overline{g}_{i+k}, m_{\overline{g}_{i+k}}] \oplus [g_{i+1}, m_{g_{i+1}}]$ can be excluded from further consideration. The stronger implication is only valid if $[g_{i+1}, m_{g_{i+1}}]$ can be locally left-shifted to start at a time $\overline{ST}_{g_{i+1}}$, $\overline{ST}_{g_{i+1}} = ST_{g_i} - d_{g_{i+1}, m_{g_{i+1}}}$. However, we can extend the considerations to global left-shifts (with and without mode changes) to reduce the enumeration to active schedules (cf. [26]).

The enumeration process is continued and then stopped for studying the branch [1,1], [2,1], [3,1], [6,1], [4,1] with its illustration presented in Figure 9. Since [4,1] can be started at $\overline{ST}_4 = 4$ without violating the constraints, we can free resources in periods t, $t = ST_4 + 1 = 14, \ldots, 18$, as a result of

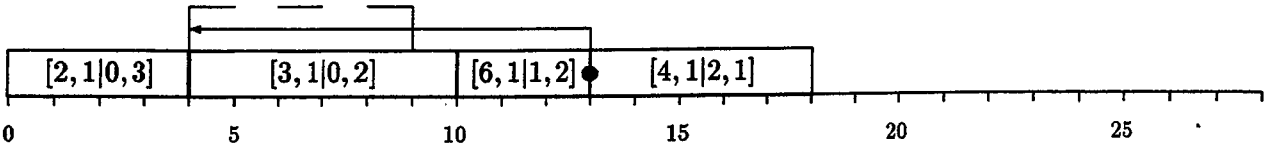

Figure 9: Global Left-Shift Rule

which the availability of the resources for scheduling the remaining activities is increased. We include mode changes and summarize as follows:

Theorem 8 *(Bounding Rule 6, Global and Multi-Mode Left-Shift Rule)*
Let $\mathcal{PS}_{i+1} = \mathcal{PS}_i \oplus [g_{i+1}, m_{g_{i+1}}]$ *be a feasible* $(i+1)$*-partial schedule and let* $ST_{g_{i+1}}$ *denote the start time of activity* g_{i+1} *scheduled in mode* $m_{g_{i+1}}$ *in* $\mathcal{PS}_{i+1}$ *and* $CT_{g_{i+1}}$ *the corresponding completion time. If, by ignoring condition (a) of Theorem 1, the i-partial schedule* $\mathcal{PS}_i$ *can be feasibly extended by scheduling a job/mode combination* $[g_{i+1}, \overline{m}_{g_{i+1}}]$ *with a minimal start time* $\overline{ST}_{g_{i+1}}$ *and related completion time* $\overline{CT}_{g_{i+1}}$ *such that*

$$\text{(a)}\ \overline{CT}_{g_{i+1}} \leq ST_{g_i}$$

$$\text{(b)}\ k^{\nu}_{g_{i+1},\overline{m}_{g_{i+1}},r} \leq k^{\nu}_{g_{i+1},m_{g_{i+1}},r}, \qquad r \in N$$

then $\mathcal{PS}_{i+1}$ *is dominated.*

Proof: Similar to the one of Theorem 7. □

Note, by commonly applying the local left-shift rule and the global left-shift rule (with or without mode change) only active schedules are generated. However, since the binding effect may or may not be produced by scheduling the last activity tightness (cf. [12], [23]) of the schedules cannot be guaranteed. Moreover, if a regular measure of performance, other than (a) through (d) of Section 2, has to be optimized, then a mode change is generally not allowed. Nevertheless, the assumptions of Theorem 8 can be strenghtened in order to allow a mode change.
For notational convenience, we subsequently state rules for the minimization of the project's makespan. By enlarging the assumptions they can be used for dealing with any regular measure of performance.

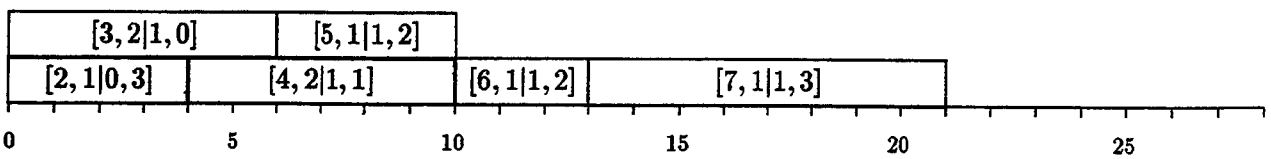

Figure 10: Feasible Solution 3

We now skip some larger portion of the branch-and-bound process. Stepping over the determination of the third feasible solution (cf. Figure 10) and the optimal solution (cf. Figure 11) we halt the enu-

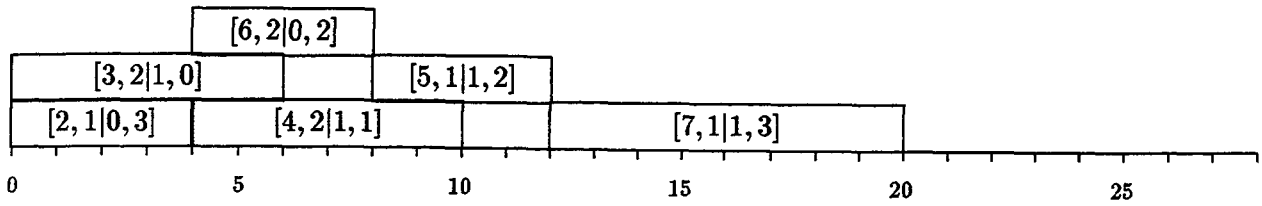

Figure 11: Feasible Solution 4 and Optimum

meration for analyzing the scheduling sequence [1,1], [2,1], [5,2], [6,2], [4,2] as displayed in Figure 12.

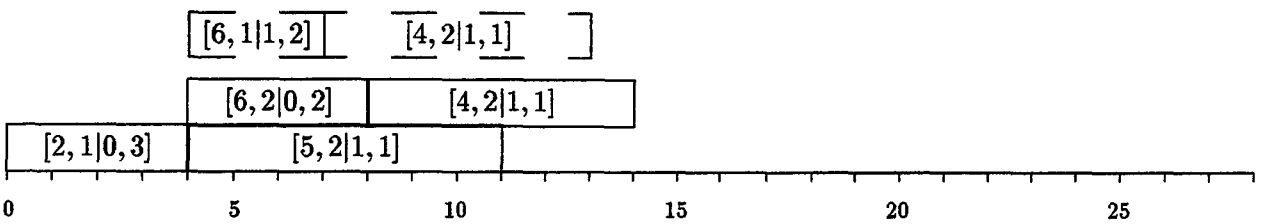

Figure 12: Multi-Mode Rule

Comparing the schedules related to the branches [1,1], [2,1], [5,2], [6,1], [4,2] and [1,1], [2,1], [5,2], [6,2], [4,2] of the precendence tree we see that in the latter partial schedule [4,2] starts at $ST_4 = 8$, that is, after the completion $\overline{CT}_6 = 7$ of [6,1] in the former one. Therefore, the same start time (or a lower one) has been feasible when scheduling [4,2] in the former partial schedule. Consequently, the left-over capacities in periods t, $t = ST_4 + 1, \ldots, \overline{T}$, of the latter schedule are at most equal to the ones of the former schedule. Using condition (a) of Theorem 1 we can summarize the results in Theorem 9 for the MRCPSP. Obviously, taking into account the partial sums of the objectives, the statement can be generalized to hold for objectives (a) through (d) of Section 2.

Theorem 9 *(Bounding Rule 7, Multi-Mode Rule)*
Let $\mathcal{PS}_{i+2} = \mathcal{PS}_i \oplus [g_{i+1}, m_{g_{i+1}}] \oplus [g_{i+2}, m_{g_{i+2}}]$ *be a feasible* $(i+2)$*-partial schedule. If there is a mode* $\overline{m}_{g_{i+1}}$, $\overline{m}_{g_{i+1}} < m_{g_{i+1}}$, *of activity* g_{i+1}, *such that the completion time* $\overline{CT}_{g_{i+1}}$ *of activity* g_{i+1} *scheduled in mode* $\overline{m}_{g_{i+1}}$ *in* $\overline{\mathcal{PS}}_{i+1} = \mathcal{PS}_i \oplus [g_{i+1}, \overline{m}_{g_{i+1}}]$ *is less than or equal to the start time* $ST_{g_{i+2}}$ *of activity* g_{i+2} *scheduled in mode* $m_{g_{i+1}}$ *in* $\mathcal{PS}_{i+2}$ *and* $k^{\nu}_{g_{i+1},\overline{m}_{g_{i+1}},r} \leq k^{\nu}_{g_{i+1},m_{g_{i+1}},r}$, $r \in N$, *then* $\mathcal{PS}_{i+2}$ *is dominated by previously evaluated continuations of* $\overline{\mathcal{PS}}_{i+2} = \mathcal{PS}_i \oplus [g_{i+1}, \overline{m}_{g_{i+1}}] \oplus [g_{i+2}, m_{g_{i+2}}]$.

Proof: Let $\overline{m}_{g_{i+1}}$ be the mode of activity g_{i+1} fulfilling the assumption of Theorem 9. Moreover, let $\overline{ST}_{g_{i+1}}$ be the start time of $[g_{i+1}, \overline{m}_{g_{i+1}}]$ in $\overline{\mathcal{PS}}_{i+1} = \mathcal{PS}_i \oplus [g_{i+1}, \overline{m}_{g_{i+1}}]$ and $\overline{CT}_{g_{i+1}}$ the corresponding completion time. We compare the left-over capacities of the $(i+2)$-partial schedule $\overline{\mathcal{PS}}_{i+2}$, $\overline{\mathcal{PS}}_{i+2} =$

$\mathcal{PS}_i \oplus [g_{i+1}, \overline{m}_{g_{i+1}}] \oplus [g_{i+2}, m_{g_{i+2}}]$ with the left-over capacities of the schedule $\mathcal{PS}_{i+2}$, $\mathcal{PS}_{i+2} = \mathcal{PS}_i \oplus [g_{i+1}, m_{g_{i+1}}] \oplus [g_{i+2}, m_{g_{i+2}}]$. Since $ST_{g_{i+2}} \geq \overline{CT}_{g_{i+1}}$, $ST_{g_{i+2}}$ is a feasible start time of activity g_{i+2} in mode $m_{g_{i+2}}$ on level $(i+2)$ with $(i+1)$-partial schedule $\overline{\mathcal{PS}}_{i+1}$. Therefore, $\overline{ST}_{g_{i+2}} \leq ST_{g_{i+2}}$, which leads to $K^{\rho}_{rt}(\mathcal{PS}_{i+2}) \leq K^{\rho}_{rt}(\overline{\mathcal{PS}}_{i+2})$ for each period t, $t = ST_{g_{i+2}}+1, \ldots, \overline{T}$, and each renewable resource r, $r \in R$, and the theorem is proven. □

The next rule we present is the multi-mode cut-set rule I. It extends in some ways earlier, only single-mode suitable, versions (cf. [6], [30]) to the multi-mode case.

Definition 5

Let $\mathcal{PS}_i$ be an i-partial schedule. The cut-set $\mathcal{CS}(\mathcal{PS}_i)$ belonging to $\mathcal{PS}_i$ is defined as the set of activities currently scheduled up to stage i, that is

$$\mathcal{CS}(\mathcal{PS}_i) \;:=\; \bigcup_{j=1}^{i} \{g_j\}$$

and the maximum related completion time $CT^{max}(\mathcal{PS}_i)$ of the activities scheduled in $\mathcal{PS}_i$ is defined by

$$CT^{max}(\mathcal{PS}_i) \;:=\; \max_{j=1}^{i} \{ST_{g_j} + d_{g_j m_{g_j}}\}.$$

Using the definition we can now illustrate, by two examples, a multi-mode version of the cut-set rule. We study the schedules $\overline{\mathcal{PS}}_5$ and $\mathcal{PS}_5$ induced by the sequence [1,1], [2,1], [3,2], [5,1], [6,1] and [1,1], [2,1], [3,2], [6,1], [5,1] (cf. Figure 13), respectivley. From the illustration we learn that interchanging of [5,1] and [6,1] does not effect (a) the set of activities currently scheduled up to level 5, i.e. $\mathcal{CS}(\overline{\mathcal{PS}}_5) = \mathcal{CS}(\mathcal{PS}_5)$, (b) the maximum completion time of the activities currently scheduled, i.e. $CT^{max}(\overline{\mathcal{PS}}_5) = CT^{max}(\mathcal{PS}_5)$, and (c) the start time of activity/mode combination [4,1] scheduled on level 6, i.e. $\overline{ST}_4 = ST_4 = 11$. We consider the left-over capacities of $\overline{\mathcal{PS}}_6$, $\overline{\mathcal{PS}}_6 = \overline{\mathcal{PS}}_5 \oplus [4,1]$ and $\mathcal{PS}_6$, $\mathcal{PS}_6 = \mathcal{PS}_5 \oplus [4,1]$ Using $K^{\rho}_{rt}(\overline{\mathcal{PS}}_6) \geq K^{\rho}_{rt}(\mathcal{PS}_6)$, $r \in R$, $t = ST_4+1, \ldots, \overline{T}$, and condition (a) of Theorem 1, we know that the completions of $\mathcal{PS}_6$ are dominated by the completions of $\overline{\mathcal{PS}}_6$.

A second example is presented in Figure 14. The partial schedules $\overline{\mathcal{PS}}_3$ and $\mathcal{PS}_3$ result from the sequences [1,1], [2,1], [3,2] and [1,1], [2,2], [3,1], respectively. Again the start time ST_5 of the activity/mode combination [5,1] currently considered for continuing the latter schedule, i.e. $\mathcal{PS}_3$, is not less than the maximum completion time of the activities 1, 2, 3 in the former schedule. Therefore, using the argumentation given above, the continuations of $\overline{\mathcal{PS}}_4 = \overline{\mathcal{PS}}_3 \oplus [5,1]$ dominate the continuations of $\mathcal{PS}_4 = \mathcal{PS}_3 \oplus [5,1]$.

The structure of the examples, the "serial" structure of the former one and the "block" structure of the latter one, illustrate that this dominance concept can be successfully used when the renewable

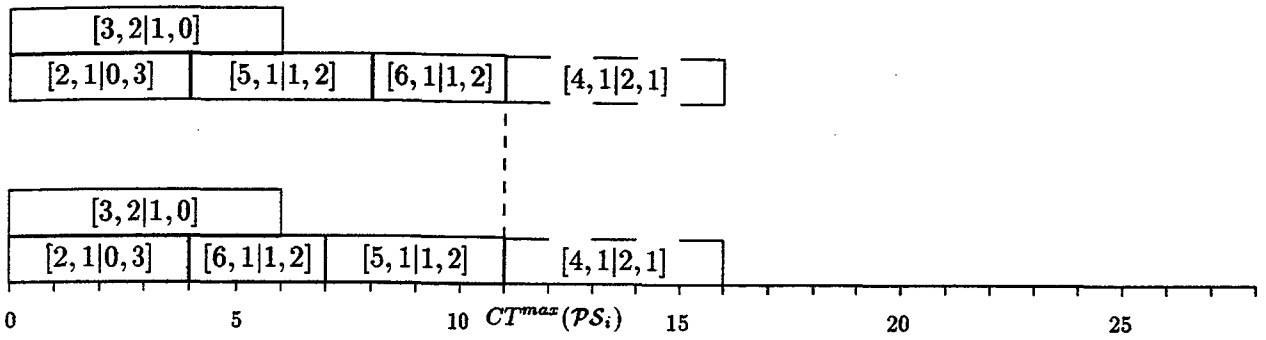

Figure 13: Multi-Mode Cut-Set Rule I – Example 1

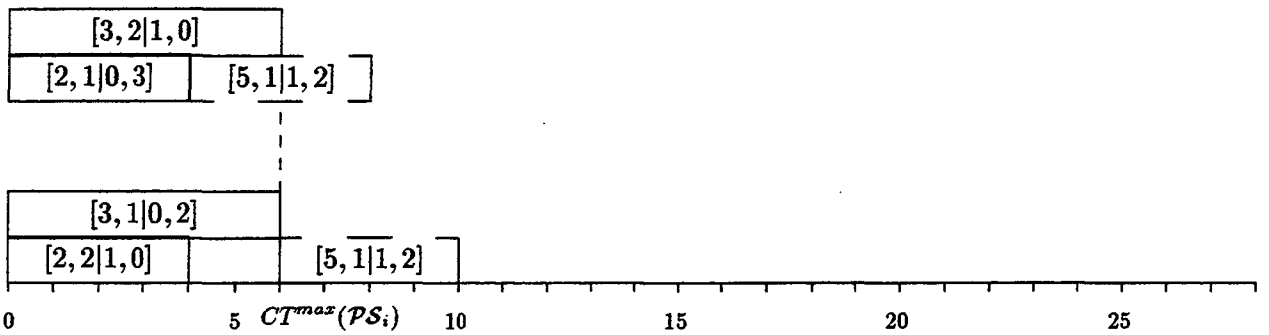

Figure 14: Multi-Mode Cut-Set Rule I – Example 2

resources are scarce or ample. We summarize the results:

Theorem 10 *(Bounding Rule 8, Multi-Mode Cut-Set Rule I, Dominated Heads)*
Let $\mathcal{PS}_i$ be the i-partial schedule currently under consideration to be extended to an $(i+1)$-partial schedule by feasibly scheduling $[g_{i+1}, m_{g_{i+1}}]$. Let $\overline{\mathcal{PS}_i}$ be an i-partial schedule with $\mathcal{CS}(\mathcal{PS}_i) = \mathcal{CS}(\overline{\mathcal{PS}_i})$ which has been stored when examing previous assignments, that is, $\overline{\mathcal{PS}_i} \neq \mathcal{PS}_i$. Then the continuations of $\mathcal{PS}_{i+1} = \mathcal{PS}_i \oplus [g_{i+1}, m_{g_{i+1}}]$ are dominated by previously evaluated continuations of $\overline{\mathcal{PS}_{i+1}} = \overline{\mathcal{PS}_i} \oplus [g_{i+1}, m_{g_{i+1}}]$ if

(a) $ST_{g_{i+1}} \geq CT^{max}(\overline{\mathcal{PS}_i})$

(b) $K_r^{\nu}(\mathcal{PS}_i) \leq K_r^{\nu}(\overline{\mathcal{PS}_i}), \qquad r \in N.$

Proof: By conditions (a) and (b) we know that $\overline{\mathcal{PS}_i}$ can be feasibly continued to $\overline{\mathcal{PS}_{i+1}} = \overline{\mathcal{PS}_i} \oplus [g_{i+1}, m_{g_{i+1}}]$ with $[g_{i+1}, m_{g_{i+1}}]$ having a start time $\overline{ST}_{g_{i+1}}$ lower or equal to the start time $ST_{g_{i+1}}$ of $[g_{i+1}, m_{g_{i+1}}]$ in $\mathcal{PS}_{i+1}$. From (b) and condition (a) of Theorem 1 we deduce

$$K_r^{\nu}(\mathcal{PS}_i \oplus [g_{i+1}, m_{g_{i+1}}]) \leq K_r^{\nu}(\overline{\mathcal{PS}_i} \oplus [g_{i+1}, m_{g_{i+1}}]), \quad r \in N$$

$$K^{\rho}_{rt}(\mathcal{PS}_i \oplus [g_{i+1}, m_{g_{i+1}}]) \leq K^{\rho}_{rt}(\overline{\mathcal{PS}_i} \oplus [g_{i+1}, m_{g_{i+1}}]), \quad r \in R, t = ST_{g_{i+1}} + 1, \ldots, \overline{T}$$

and the proof is complete. □

Note, although there is a certain similarity between the multi-mode rule (Bounding Rule 7) and the cut-set rule I (Bounding Rule 8), Figures 12 through 14 show examples where the former rule can be applied and the latter one not, and vice versa.

Moreover, only minor modifications for dealing with the performance measures (a) through (e), given in Section 2, are necessary. That is, additionally taking into account the partial schedule related partial sums the rule can be extended.

Roughly speaking, the cut-set rule I compares the quality of the current $(i+1)$-partial schedule with the quality of an $(i+1)$-partial schedule previously evaluated. To distinguish it form the following cut-set based rule we call it dominated heads. In contrast, the last rule we are going to present bounds the enumeration process by estimating the time necessary to complete the partial schedule under consideration. That is, it offers a criterion for incompletable tails. We assume the resource availability of the renewable resources as constant and the minimization of the makespan as objective.

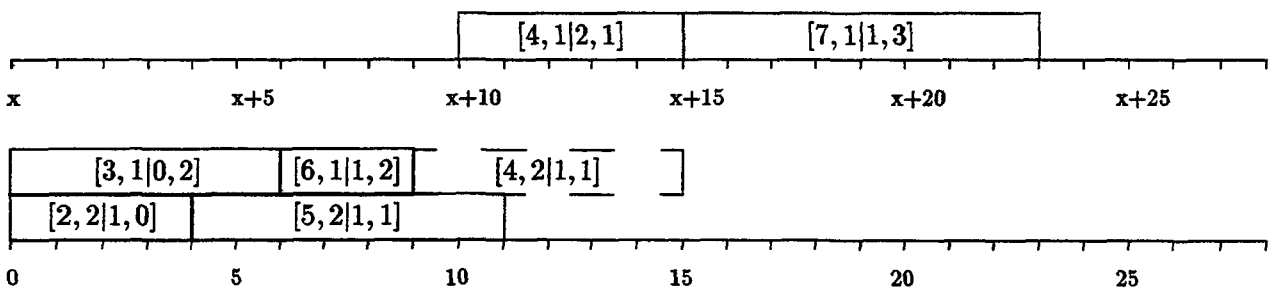

Figure 15: Multi-Mode Cut-Set Rule II

We observe the partial schedule given in Figure 15, where activties 1, 2, 3, 5, and 6, i.e. $\mathcal{CS}(\mathcal{PS}_5) = \{1,2,3,5,6\}$ are scheduled. Activities 4, 7, and 8 are unscheduled. We are now trying to schedule activity 4 in mode 1 (2) and determine the start time $ST_4 = 9$. The upper part of the graphic shows, that, if no activity is processed in the periods activities 4, 7, 8 have to be scheduled in, that is, no activity uses resources in the periods considered, then it takes (at least) 13 periods to finish the unscheduled activities. Since $ST_4 + 13 > LF_J = 19$, the partial schedule cannot be completed with a makespan less than or equal to LF_J and the related branch can be truncated. The following rule says how to get the minimal necessary prolongation for the completion of the partial schedule. The data

required is obtained as a byproduct of cut-set rule I.

Theorem 11 *(Bounding Rule 9, Multi-Mode Cut-Set Rule II, Incompletable Tails)*
We consider a problem of type MRCPSP with constant availabilities of the renewable resources. Moreover, let $\mathcal{PS}'_k$ be any k-partial schedule. By $LF_J(\mathcal{PS}'_k)$ we denote the latest finish time of activity J valid after all the continuations of $\mathcal{PS}'_k$ have been evaluated by the enumeration scheme given in **Table 4**. *That is, the bounds on the finish times have been adjusted after finding an improved solution. Let $\mathcal{PS}_i$ be the i-partial schedule currently under consideration to be continued to an $(i+1)$-partial schedule by feasibly scheduling $[g_{i+1}, m_{g_{i+1}}]$. Let $\overline{\mathcal{PS}_i}$ be an i-partial schedule with $CS(\mathcal{PS}_i) = CS(\overline{\mathcal{PS}_i})$ which has been stored when examining previous assignments, that is, $\overline{\mathcal{PS}_i} \neq \mathcal{PS}_i$. If*

(a) $K_r^v(\mathcal{PS}_i) \leq K_r^v(\overline{\mathcal{PS}_i}), \qquad r \in N,$

(b) $ST_{g_{i+1}} + LF_J(\overline{\mathcal{PS}_i}) - CT^{max}(\overline{\mathcal{PS}_i}) + 1 > LF_J,$

then $\mathcal{PS}_{i+1} = \mathcal{PS}_i \oplus [g_{i+1}, m_{g_{i+1}}]$ cannot be completed with a makespan at most equal to the currently valid late finish time LF_J of activity J.

Proof: We suppose the statement is wrong, that is, although the assumptions hold, there is a continuation $\mathcal{PS}_J$ of $\mathcal{PS}_{i+1}$ with

$$\mathcal{PS}_J = \mathcal{PS}_i \oplus [g_{i+1}, m_{g_{i+1}}] \oplus \cdots \oplus [g_J, m_{g_J}] \tag{7}$$

and $CT^{max}(\mathcal{PS}_J) \leq LF_J \leq LF_J(\overline{\mathcal{PS}_i})$. Due to assumption (a) the modes selected in (7) can be chosen to continue $\overline{\mathcal{PS}_i}$ without violating the nonrenewable resource constraints. None of the activities scheduled in $\overline{\mathcal{PS}_i}$ is in process after $CT^{max}(\overline{\mathcal{PS}_i})$, that is, we can use the sequence $[g_{i+1}, m_{g_{i+1}}], \ldots, [g_J, m_{g_J}]$ to complete $\overline{\mathcal{PS}_i}$ to a schedule $\overline{\mathcal{PS}_J}$, $\overline{\mathcal{PS}_J} = \overline{\mathcal{PS}_i} \oplus [g_{i+1}, m_{g_{i+1}}] \oplus \cdots \oplus [g_J, m_{g_J}]$ with $CT^{max}(\overline{\mathcal{PS}_J}) \leq LF_J(\overline{\mathcal{PS}_i})$. This contradicts the definition of $LF_J(\overline{\mathcal{PS}_i})$. □

4.3 Commonly Applying the Dynamic Rules

We close this section with some comments on commonly using the dynamic search tree reduction techniques presented above. Recall, Bounding Rule 3 identifies non-delayable activities, Bounding Rule 4 enforces single enumeration, Bounding Rule 5 excludes non-semi-active schedules, Bounding Rule 6 detects feasible global and multi-mode left-shifts, Bounding Rule 7 considers mode reductions, Bounding Rule 8 excludes dominated heads (multi-mode cut-set rule I), and Bounding Rule 9 prevents from continuing incompletable tails (multi-mode cut-set rule II). We assume the cut-sets related to an i-partial schedule to be stored when tracking back from level $(i+1)$ to level i.

Bounding Rule 3 (non-delayability): Clearly, when combining the non-delayability rule with the *single enumeration*, the *local left-shift rule*, the multi-mode rule, the multi-mode cut-set rule I or the multi-mode cut-set rule II the job/mode combinations excluded from consideration on the current level by one of the latter rules cannot be considered as not schedulable. The delay of these activity/mode combinations might produce an optimal solution. In contrast job/mode combinations, that are excluded with respect to the global and multi-mode left-shift rule need not be delayed to a higher level. The same left-shifts could be applied to a continuation using that specific job/mode combination.

Bounding Rule 4 (single enumeration): Combining the Bounding Rule 4 with any other rule, except of the non-delayability rule, is not critical, since the rule does not really exclude schedules from enumeration. It only prevents duplicate consideration of the same schedules related to different sequences.

Bounding Rule 5 and 6 (shift rules): Ignoring the non-delaybility rule and the multi-mode cut-set rule II, the local, global and multi-mode left-shifts can be applied together with any other rule. They only exclude partial schedules from continuation that are dominated by semi-active or active schedules. Worth mentioning, by applying the shift rules the cut-sets of the related i-partial schedules differ from the one of the current i-partial schedule. That is, the cut-set storing policy prevents that an $(i+1)$-partial schedule excluded from continuation due to a shift-rule can be used within the cut-set rule to exclude the $(i+1)$-partial schedule derived by the shift. However, it might happen that the current schedule is optimally continued by scheduling a left-shiftable job/mode combination on the current level and continuing the $(i+1)$-partial schedule obtained. That is, by cut-set rule II an estimation of the minimum time necessary to complete an i-partial schedule with same cut-set and left-over capacities of nonrenewable resources at most equal to the ones of the currently considered schedule cannot be obtained if activities are left-shiftable.

Bounding Rule 7 (mode reduction): When combining the shift-rules, the multi-mode and the cut-set rule I one has to take into account that backtracking induced cut-set storing may produce an error. That is, if $[g_{i+1}, m_{g_{i+1}}]$ schedulable with completion time $CT_{g_{i+1}}$ and left-shiftable and, moreover, the current combination, $[g_{i+1}, \overline{m}_{g_{i+1}}]$, $\overline{m}_{g_{i+1}} > m_{g_{i+1}}$, consumes at most the same amounts of the nonrenewable resources as $[g_{i+1}, m_{g_{i+1}}]$ does and, moreover, the start time of an activity/mode combination $[g_{i+2}, m_{g_{i+2}}]$ is equal to or higher than $CT_{g_{i+1}}$ then the truncation of the current branch may cause suboptimality of the solution obtained. To overcome the problem, one simply has to exclude bounding by the multi-mode rule using the completion time of a left-shiftable job/mode combination.

Bounding Rule 8 (multi-mode cut-set rule I) and 9 (multi-mode cut-set rule II): As already mentioned above combining the shift-rules with the cut-set-rule II may produce subobtimality if the above named facts are not considered. However, although, we can handle the problem by storing the minimum time necessary to complete $\mathcal{PS}_i$, as given in Theorem 11 (b), only if none of the activity/mode combinations $[g_{i+1}, m_{g_{i+1}}]$, $g_{i+1} \in Y_{i+1}$, $1 \leq m_{g_{i+1}} \leq M_{g_{i+1}}$, can be left-shifted, the additional use of cut-set rule I may cause suboptimality. For illustrating, that the proof of Theorem 11 cannot be adapted, though the extended assumptions hold, we consider the project given in Figure 16. Obviously, the problem has an optimal solution with a makespan equal to the MPM-

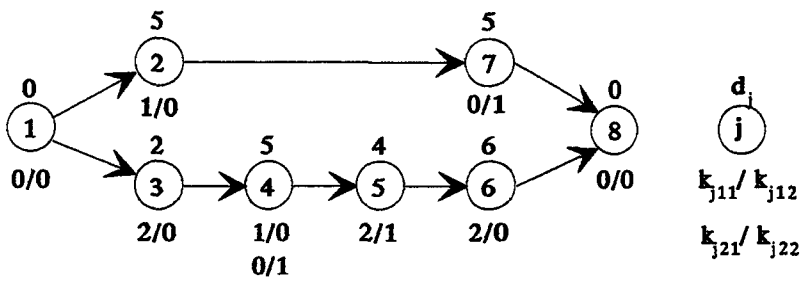

Figure 16: Example Project – $K_1^\rho = 2$, $K_2^\rho = 1$

duartion of 17 periods. It is obtained by the sequence [1,1], [3,1], [2,1], [4,1] ([4,2]), [5,1], [6,1], [7,1], [8,1]. We start the enumeration process with $\overline{T} = 27$. Using the branch [1,1], [2,1], [3,1], [4,1], we see that it is not completed, if the shift rules are employed. On the other hand we obtain the cut-set $\mathcal{CS}(\mathcal{PS}_4) = \{1,2,3,4\}$ with $CT^{max}(\mathcal{PS}_4) = 12$. It serves for truncating the continuations of $\overline{\mathcal{PS}_4} = [1,1] \oplus [2,1] \oplus [3,1] \oplus [4,2]$. Note, none of the activity/mode combinations eligible on level 5 can be left-shifted. The minimum time neccessary to complete a partial schedule with same cut-set as $\overline{\mathcal{PS}_4}$ would be determind as $LF_J(\overline{\mathcal{PS}_4}) - CT^{max}(\overline{\mathcal{PS}_4}) + 1 = (27 - 12 + 1) = 16$. The first feasible solution is derived by the sequence [1,1], [2,1], [3,1], [7,1], [4,1], [5,1], [6,1], [8,1] leading to a makespan of 22 periods, that is $LF_8 = 21$. Continuing the sequence [1,1], [3,1], [2,1], [4,1], we determine start times $ST_5 = ST_7 = 7$ on level 5. Using the (wrong) bound the continuations of the sequence are excluded by cut-set rule II. Subsequently, the continuations of the sequence [1,1], [3,1], [2,1], [4,2] are excluded by cut-set rule I.

Therefore, although the theoretical consideration is of interest, in our implementation we combine the multi-mode cut-set rule II only with the cut-set rule I. Otherwise we prefer the beneficial combination of the cut-set rule I and the shift rules.

5 Conclusions

We have discussed an enumeration scheme for solving the multi-mode resource-constrained project scheduling problem. The basic scheme can be easily generalized for additionally handling constraints induced by time varying requests as well as minimal and maximal time lags. Moreover, any objective considered can be dealt with a slightly modified version. Using i-partial schedules the enumeration scheme and the current state of the enumeration process can be easily described and extended by search tree reduction schemes.

In the second part of the paper we present the results of our thorough computational experiments. Although, for dealing with resource-constrained project scheduling problems, the (slightly modified version of the) algorithm is the most general one, the exact method shows surperior performance when solving the multi-mode resource-constrained project scheduling problem. In addition, the truncated exact method turns out to be a reasonable heuristic solution strategy.

References

[1] BAKER, K.R. (1974): Introduction to sequencing and scheduling, Wiley, New York et al.

[2] BARTUSCH, M.; R.H. MÖHRING AND F.J. RADERMACHER (1988): Scheduling project networks with resource constraints and time windows. Annals of Operations Research, Vol. 16, pp. 201-240.

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

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

[5] DAVIS, E.W. AND G.E. HEIDORN (1971): An algorithm for optimal project scheduling under multiple resource constraints. Management Science, Vol. 17, pp. B803-B816.

[6] DEMEULEMEESTER, E. AND W. HERROELEN (1992): A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. Management Science, Vol. 38, pp. 1803-1818.

[7] DREXL, A. (1991): Scheduling of project networks by job assignment. Management Science, Vol. 37, pp. 1590-1602.

[8] DREXL, A. AND J. GRÜNEWALD (1993): Nonpreemptive multi-mode resource-constrained project scheduling. IIE Transactions, Vol. 25, No. 5, pp. 74-81.

[9] ELMAGHRABY, S.E. (1977): Activity networks: Project planning and control by network models. Wiley, New York.

[10] FRENCH, S. (1982): Sequencing and scheduling: An Introduction to the mathematics of the job-shop. Wiley, New York.

[11] GAREY, M.R. AND D.S. JOHNSON (1979): Computers and intractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisco, CA.

[12] HARTMANN, S.; A. SPRECHER (1995): A note on "hierarchical models for multi-project planning and scheduling". To appear in: European Journal of Operational Research

[13] KOLISCH, R. (1995): Project scheduling under resource constraints: Efficient heuristics for several problem classes. Physica-Verlag, Heidelberg.

[14] KOLISCH, R.; A. SPRECHER AND A. DREXL (1995): Characterization and generation of a general class of resource-constrained project scheduling problems. Management Science, Vol. 41, No. 11.

[15] 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.

[16] PATTERSON, J.H.; R. SLOWINSKI; F.B. TALBOT AND J. WEGLARZ (1990): Computational experience with a backtracking algorithm for solving a general class of precedence and resource-constrained scheduling problems. European Journal of Operational Research, Vol. 49, pp. 68-79.

[17] RADERMACHER, F.J. (1985/86): Scheduling of project networks. Annals of Operations Research, Vol. 4, pp. 227-252.

[18] RINNOOY KAN, A.H.G. (1976): Machine scheduling problems: Classification, complexity and computation. Nijhoff, The Hague.

[19] SLOWINSKI, R. (1980): Two approaches to problems of resource allocation among project activities: A comparative study. Journal of the Operational Research Society, Vol. 31, pp. 711-723.

[20] SLOWINSKI, R. (1981): Multiobjective network scheduling with efficient use of renewable and nonrenewable resources. European Journal of Operational Research, Vol. 7, pp. 265-273.

[21] 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. 151-167.

[22] SLOWINSKI, R.; B. SONIEWICKI AND J. WEGLARZ (1994): DSS for multiobjective project scheduling. European Journal of Operational Research, Vol 79, pp. 220-229.

[23] SPERANZA, M.G. AND C. VERCELLIS (1993): Hirarchical models for multi-project planning and scheduling. European Journal of Operational Research, Vol. 64, pp. 312-325.

[24] SPRECHER, A. (1994): Resource-constrained project scheduling: Exact methods for the multi-mode case. Lecture Notes in Economics, No. 409. Springer, Berlin.

[25] SPRECHER, A.; S. HARTMANN AND A. DREXL (1994): Project scheduling with discrete time-resource and resource-resource tradeoffs. Manuskripte aus den Instituten für Betriebswirtschaftslehre, No. 357, Kiel.

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

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

[28] TALBOT, F.B. (1980): Project scheduling with resource-duration interactions: The nonpreemptive case. Working Paper, The Graduate School of Business Administration, University of Michigan, USA.

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

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

[31] WEGLARZ, J. (1979): Project scheduling with discrete and continuous resources. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 9, pp. 644-650.

[32] WEGLARZ, J. (1980): On certain models of resource allocation problems. Kybernetics, Vol. 9, pp. 61-66.