1-1999

# Algorithms for scheduling projects with generalized precedence relations

Bert DE REYCK
*Singapore Management University*, bdreyck@smu.edu.sg

Erik DEMEULEMEESTER

Willy HERROELEN

# 4 ALGORITHMS FOR SCHEDULING PROJECTS WITH GENERALIZED PRECEDENCE RELATIONS

Bert De Reyck[1]
Erik Demeulemeester[2]
Willy Herroelen[2]

*1. Erasmus University Rotterdam (The Netherlands)*
*2. Katholieke Universiteit Leuven (Belgium)*

## 4.1. Introduction

The problem of scheduling projects under various types of resource constraints constitutes an important and challenging problem which has received increasing attention during the past several years. The bulk of the models and procedures designed for coping with these problem types aim at scheduling project activities to minimize the project duration subject to constant availability constraints on the required set of resources and precedence constraints that indicate that activities can only be started when all of their predecessors have already been finished. However, real-life project scheduling applications often involve more complicated types of precedence relations such as arbitrary minimal and maximal time lags between the starting and completion times of the activities, and require more sophisticated regular and nonregular objective functions. Over the past few

years, considerable progress has been made in the use of exact solution procedures for this problem type and its variants. We will review the fundamental logic and report new computational experience with solution procedures for optimally solving resource-constrained project scheduling problems in which such generalized precedence relations and objective functions can be explicitly considered.

We distinguish between four types of generalized precedence relations (GPRs): start-start (SS), start-finish (SF), finish-start (FS) and finish-finish (FF). These relations specify a minimal or maximal time lag between a pair of activities. A minimal time lag specifies that an activity $j$ can only start (finish) when its predecessor $i$ has already started (finished) for a certain time period. A maximal time lag specifies that an activity should be started (finished) at the latest a specific number of time periods beyond the start (finish) of activity $i$.

GPRs enhance the capabilities of project scheduling models because they can be used to model a wide variety of real-life problem characteristics. Next to the straightforward use of GPRs, namely allowing activity overlaps (which will often lead to a substantial decrease of the project makespan) and ensuring a maximal delay between the execution of specific activities (useful, for instance, when dealing with perishable products or chemical operations), GPRs can be used to model a wide variety of specific problem characteristics, including activity release dates and deadlines, activities that have to start or terminate simultaneously, non-delay execution of activities, several types of mandatory activity overlaps, fixed activity start times, time-varying resource requirements and availabilities, set-up times, overlapping production activities (process batches, transfer batches) and assembly line zoning constraints.

The first comprehensive treatment of GPRs is due to Kerbosch and Schell (1975), based on the pioneering work of Roy (1962). Other studies include Crandall (1973), Elmaghraby (1977), Wiest (1981), Moder et al. (1983), Bartusch et al. (1988), Elmaghraby and Kamburowski (1992), Zhan (1994), Neumann and Zhan (1995), Schwindt (1996), Brinkmann and Neumann (1996), Schwindt and Neumann (1996), Franck and Neumann (1996), De Reyck and Herroelen (1996b, 1997, 1998ab), Neumann and Schwindt (1997), Demeulemeester and Herroelen (1997b) and De Reyck (1998).

The remainder of this chapter is organized as follows. Section 4.2 elaborates on the concept of GPRs and clarifies the terminology and project representation used. Section 4.3 briefly reviews the temporal analysis of activity networks with GPRs. A distinction is made between the precedence diagramming case, i.e. start-start, start-finish, finish-start and finish-finish

relations with minimal time lags for minimizing the project makespan (defined as $min|C_{max}$ using the classification scheme of Herroelen et al. 1998), and the case of minimal and maximal time lags ($gpr|C_{max}$). In Section 4.4, the resource analysis required by the introduction of additional resource constraints is discussed. In Section 4.5, we discuss the fundamentals of a branch-and-bound procedure for optimally solving resource-constrained project scheduling problems with generalized precedence relations of the precedence diagramming type. This problem type, when extended to cope with activity release dates and deadlines as well as variable resource availabilities, is also referred to as the *generalized resource-constrained project scheduling problem* (Demeulemeester and Herroelen 1997b) and is denoted as $m,1,va|min,\rho_i,\delta_i|C_{max}$ using the classification scheme of Herroelen et al. (1998). Subsequently in Section 4.6, we review several branch-and-bound procedures for the case of minimal and maximal time lags ($m,1|gpr|C_{max}$), and demonstrate how the solution methodology can be extended to cope with variable resource availabilities and requirements as well as other real-life project characteristics ($m,1,va|gpr,\rho_i,\delta_i,vr|C_{max}$), and with other regular ($m,1,va|gpr,\rho_i,\delta_i,vr|reg$) and nonregular ($m,1,va|gpr,\rho_i,\delta_i,vr|nonreg$) objective functions. In Section 4.7, we briefly describe the modifications that the original algorithms have undergone since their development. In Section 4.8, computational experience is reported using a set of randomly generated problem instances. Section 4.9 is reserved for our conclusions.

## 4.2. Terminology and representation

Assume a project represented in activity-on-the-node format by a directed graph $G = \{V, E\}$ in which $V$ is the set of vertices or activities, and $E$ is the set of edges or GPRs. The non-preemptable activities are numbered from 1 to $n$, where the dummy activities 1 and $n$ mark the beginning and the end of the project. The duration of an activity is denoted by $d_i (1 \le i \le n)$, its start time by $s_i (1 \le i \le n)$ and its finish time by $f_i (1 \le i \le n)$. There are $m$ renewable resource types, with $r_{ikx} (1 \le i \le n, 1 \le k \le m, 1 \le x \le d_i)$ the resource requirements of activity $i$ with respect to resource type $k$ in the $x^{th}$ period it is in progress and $a_{kt} (1 \le k \le m; 1 \le t \le T)$ the availability of resource type $k$ in time period $]t\text{-}1, t]$ ($T$ is an upper bound on the project length). If the resource requirements and availabilities are not time-dependent, they are represented by $r_{ik} (1 \le i \le n, 1 \le k \le m)$ and $a_k (1 \le k \le m)$ respectively. The minimal and maximal time lags between two activities $i$ and $j$ have the form:

$$s_i + SS_{ij}^{\min} \le s_j \le s_i + SS_{ij}^{\max}; \qquad s_i + SF_{ij}^{\min} \le f_j \le s_i + SF_{ij}^{\max}$$
$$f_i + FS_{ij}^{\min} \le s_j \le f_i + FS_{ij}^{\max}; \qquad f_i + FF_{ij}^{\min} \le f_j \le f_i + FF_{ij}^{\max}$$

where $SS_{ij}^{\min}$ represents a minimal time lag between the start time of activity $i$ and the start time of activity $j$ (similar definitions apply for $SS_{ij}^{\max}$, $FS_{ij}^{\min}$,...). The various time lags can be represented in a *standardized form* by transforming them to minimal start-start precedence relations, using the transformation rules given in Bartusch et al. (1988). Consequently, all GPRs are consolidated in the expression $s_i + l_{ij} \le s_j$, where $l_{ij}$ denotes a minimal start-start time lag.

A path $<i_s, i_k, i_l, ..., i_t>$ is called a *cycle* if $s = t$. With 'path' we mean a *directed* path, and with 'cycle' we mean a *directed* cycle. The *length* of a path (cycle) is defined as the sum of the lags associated with the arcs belonging to that path (cycle). To ensure that the dummy start and finish activities correspond to the start and the completion of the project, we assume that there exists at least one path with nonnegative length from node 1 to every other node and at least one path from every node $i$ to node $n$ which is equal to or larger than $d_i$. If there are no such paths, we can insert arcs $(1,i)$ or $(i,n)$ with weight zero or $d_i$ respectively. $P(i) = \{j \mid (j,i) \in E\}$ is the set of all *immediate predecessors* of node $i$, $Q(i) = \{j \mid (i,j) \in E\}$ is the set of all its *immediate successors*.

## 4.3. Temporal analysis

The (resource-unconstrained) project scheduling problem with GPRs under the minimum makespan objective ($gpr|C_{max}$) can be mathematically formulated as follows:

$$\textit{Minimize } s_n \qquad [1]$$

*Subject to*

$$s_i + l_{ij} \le s_j \qquad (i,j) \in E \qquad [2]$$
$$s_i \in N \qquad i \in V \qquad [3]$$

where N denotes the set of natural numbers. The objective function [1] minimizes the project duration (makespan), determined by the completion time (or start time, since $d_n = 0$) of the dummy end activity $n$. Constraints [2] represent the GPRs. Constraints [3] ensure that all start times assume nonnegative integer values. Solving this problem can be accomplished by

finding a precedence-feasible *earliest start schedule* (*ESS*), i.e. the minimum start times ($es_1$, $es_2$, ..., $es_n$) satisfying [2] and [3]. The earliest start of an activity $i$ can be computed by finding the longest path from node 1 to node $i$.

### 4.3.1. The precedence diagramming case

The CPM analysis for project networks with zero-lag finish-start precedence relations ($cpm|C_{max}$) can easily be extended to cope with minimal time lags ($min|C_{max}$). A forward computation step $es_i = \max\{es_j + l_{ji} | \forall j \in P(i)\}$ yields an *ESS* (assuming that $es_1 = 0$). A backward computation step $ls_i = \min\{ls_j - l_{ij} | \forall j \in Q(i)\}$ (assuming that $ls_n = es_n$) yields a latest start schedule (*LSS*) which can be used for float calculations and activity criticality analysis.

### 4.3.2. The case of generalized precedence relations

When maximal time lags are introduced ($gpr|C_{max}$), there may not be a schedule that satisfies all of the GPRs. There exists a precedence-feasible schedule for $G$ iff $G$ has no cycle of positive length (Bartusch et al. 1988). Therefore, if we compute the matrix $D = [d_{ij}]$, where $d_{ij}$ denotes the longest path from node $i$ to node $j$, a positive path length from node $i$ to itself ($d_{ii}>0$) indicates the existence of a cycle of positive length and, consequently, the non-existence of a precedence-feasible schedule. The computation of the matrix $D$ can be done by the Floyd-Warshall algorithm in time $O(n^3)$ (see Lawler 1976). The $ESS = (es_1, es_2, ..., es_n)$ is given by $(d_{11}, d_{12}, ..., d_{1n})$.

## 4.4. Resource analysis

When we introduce additional renewable resource constraints, we obtain problems $m,1|min|C_{max}$ and $m,1|gpr|C_{max}$, which can both be conceptually formulated as follows:

$$\text{Minimize } s_n \qquad [4]$$

*Subject to*

$$s_i + l_{ij} \le s_j \qquad \forall (i,j) \in E \qquad [5]$$

$$\sum_{i \in S(t)} r_{ik} \le a_k \qquad k = 1,2,...,m \quad t = 1,2,...,T \qquad [6]$$

$$s_i \in N \qquad i = 1,2,...,n \qquad [7]$$

where $S(t)$ is the set of activities in progress in time period $]t\text{-}1, t]$ and $T$ is an upper bound on the project duration, for instance $T = \sum_{i \in V} \max\left\{ d_i, \max_{j \in Q(i)} \left\{ l_{ij} \right\} \right\}$. The objective function [4] minimizes the project duration. The GPRs are denoted in standardized form by constraints [5]. Constraints [6] represent the resource constraints and constraints [7] ensure that the activity start times assume nonnegative integer values.

In the precedence diagramming case, the $l_{ij}$ values are restricted to nonnegative values. Consequently, an activity can never start before its predecessor. Activity release dates $\rho_i$ need not be specified separately because they can be modelled using standardized time lags of the type $l_{1i} = \rho_i$. The algorithm of Demeulemeester and Herroelen (1997b) for the generalized resource-constrained project scheduling problem (GRCPSP; $m,1,va|min,\rho_i,\delta_i|C_{max}$), also deals with deadlines $\delta_i$ and variable resource availabilities.

In the case of GPRs, the $l_{ij}$ values may assume arbitrary integer values. In that case also, there is no need to specify activity deadlines separately, because they can be modelled using negative standardized time lags of the type $l_{i1} = d_i - \delta_i$. Also time-varying resource availabilities and requirements need not be specified explicitly (Bartusch et al. 1988). Time-varying resource availabilities can be handled by creating dummy activities which absorb a certain amount of each resource type for which a constant availability (equal to the maximum availability over time of that resource type) can then be assumed. These dummy activities should then be assigned a fixed start time using appropriate minimal and maximal time lags. Time-varying resource requirements can be modelled by splitting up the activities in a number of subactivities with a different *constant* resource requirement for each of the resource types. The subactivities should then be connected with appropriate minimal and maximal time lags which ensure a non-delay execution of all the subactivities of each activity. Therefore, problem $m,1,va|gpr|C_{max}$ and problem $m,1,va|gpr,\rho_i,\delta_i,vr|C_{max}$ can be solved using the same algorithm.

Problems $m,1|min|C_{max}$ and $m,1|gpr|C_{max}$ are known to be strongly NP-hard. For problem $m,1|gpr|C_{max}$ or problem $m,1|min,\delta_n|C_{max}$ with an imposed project deadline, even the decision problem of detecting problem (in)feasibility is NP-complete (Bartusch et al. 1988). To the best of our knowledge, the only exact solution procedures presented in the literature are the branch-and-bound algorithms of Bartusch et al. (1988), Demeulemeester and Herroelen (1997b) and De Reyck and Herroelen (1998a). Because of the extreme complexity of problem $m,1|gpr|C_{max}$, quite a number of

heuristics have been developed (Zhan 1994, Neumann and Zhan 1995, Brinkmann and Neumann 1996, Franck and Neumann 1996, Schwindt and Neumann 1996).

In the next section, we will review two exact solution procedures for project scheduling problems with GPRs. Again, we distinguish between the precedence diagramming case $(m,1,va|min,\rho_i,\delta_i|C_{max})$, for which the procedure of Demeulemeester and Herroelen (1997b) will be reviewed, and the case of generalized precedence relations $(m,1,va|gpr,\rho_i,\delta_i,vr|C_{max})$, for which the fundamentals of the procedure developed by De Reyck and Herroelen (1998a) will be discussed.

## 4.5. A branch-and-bound procedure for the GRCPSP

### *4.5.1. The search tree*

The branch-and-bound procedure of Demeulemeester and Herroelen (1997b, further referred to as GDH) is an extension of the DH-algorithm presented in Demeulemeester and Herroelen (1992,1997a) for the resource-constrained project scheduling problem with zero-lag finish-start precedence constraints $(m,1|cpm|C_{max})$. It is based on a depth-first solution strategy in which nodes in the search tree represent resource- and precedence-feasible partial schedules. Branches emanating from a parent node correspond to exhaustive and subset-minimal combinations of activities, the delay of which resolves a resource conflict at the parent node (referred to as *minimal delaying alternatives*). The search process closely resembles the one used in the procedure of Demeulemeester and Herroelen (1992) for the RCPSP $(m,1|cpm|C_{max})$. The modifications involve a different definition of the decision point, a different process of delaying temporarily scheduled activities, a different definition of the cutset activities, and a modified backtracking scheme. In addition, the procedure relies on a different set of dominance rules and bounding calculations.

The nodes in the search tree correspond to partial schedules in which finish times temporarily have been assigned to a subset of the activities of the project. Scheduling decisions are temporary in the sense that scheduled activities may be delayed as the result of decisions made at later stages in the search process. Partial schedules are constructed by semi-active timetabling[1]; i.e. each activity is started as soon as possible within the

[1] Sprecher and Drexl (1996) correctly claim that the procedure of Demeulemeester and Herroelen (1992) does not only generate semi-active schedules. The same applies to the algorithm described here. Only if the left-shift dominance rule is extended to examine also local left-shifts prior to the current decision point, the schedules are guaranteed to be semi-

precedence and resource constraints. A precedence-based lower bound is calculated by adding the maximal remaining critical path length of any of the activities that belong to a delaying alternative to the current delaying point. The delaying alternative with the smallest lower bound is chosen for further branching. When a complete schedule is constructed or when a partial schedule can be dominated using one of the node fathoming rules described below, the procedure backtracks to a previous level in the search tree. The procedure is completed upon backtracking to level 0. Activity deadlines are coped with through a standard critical path-based backward pass computation starting from the deadlines.

### *4.5.2. Node fathoming rules*

Three dominance rules are used to prune the search tree. Proofs can be found in Demeulemeester and Herroelen (1997b).

***THEOREM 1.*** *In order to resolve a resource conflict it is sufficient to consider only minimal delaying alternatives (which do not contain other delaying alternative as a subset).*

***THEOREM 2.*** *Consider a partial schedule $PS_t$ (the set of scheduled or completed activities) at level p of the search tree in which activity i is started at time t. If activity i was delayed at level p-1 of the search tree, and if this activity can be left-shifted without violating the precedence or resource constraints, then the partial schedule $PS_t$ is dominated.*

***THEOREM 3.*** *Consider a cutset $C_t$ (the set of unscheduled activities for which at least one direct predecessor belongs to $PS_t$) which contains the same activities as a cutset $C_{t'}$, which was previously saved during the search of another path in the search tree, and which was considered during the same resource interval. If $t' \leq t$, if all activities in progress at time t' did not finish later than the maximum of t and the finish time of the corresponding activities in $PS_t$, and if the earliest possible start time of every activity in $C_{t'}$ is smaller than or equal to the earliest start time of the corresponding activity in $C_t$, then the current partial schedule $PS_t$ is dominated.*

Remark here that the definition of the cutset differs slightly from the one formulated in Demeulemeester and Herroelen (1997b). In the GRCPSP, contrary to the RCPSP, it is possible that an unscheduled activity is precedence constrained by an already scheduled activity, while some of its

---

active. However, the branching scheme and the dominance rules are based on the *principle* of semi-active timetabling.

other predecessors are not yet scheduled. A subtle change in the definition of the cutset is therefore needed. This fact was overlooked in the original implementation of the algorithm. Based on new computational experience presented in this chapter, we discovered one project example (out of the many examined) for which the optimal solution was missed because of this flaw. As will be indicated in Section 4.8.2.3, the correction of the flaw only slightly affects the computational results.

## 4.6. A branch-and-bound procedure for the RCPSP-GPR

### *4.6.1. The search tree*

Essentially, the algorithm of De Reyck and Herroelen (1998a, further referred to as DRH) is a hybrid depth-first / laser beam search branch-and-bound algorithm. The nodes in the search tree represent the initial project network, described by the longest path matrix $D = [d_{ij}]$, extended with extra zero-lag finish-start precedence relations to resolve a number of resource conflicts, which results in an *extended* matrix $D' = [d'_{ij}]$. Nodes which represent precedence-feasible but resource-infeasible project networks and which are not fathomed by any node fathoming rules described below lead to a new branching. Resource conflicts are resolved using the concept of minimal delaying alternatives. However, contrary to the GDH procedure, each of these minimal delaying alternatives is delayed (enforced by extra zero-lag finish-start precedence relations $i \prec j$, implying $s_i + d_i \leq s_j$) by each of the remaining activities also belonging to the *conflict set* $S(t^*)$, the set of activities in progress in period $]t^*\text{-}1, t^*]$ (the period of the *first* resource conflict). Consequently, each minimal delaying alternative can give rise to several *minimal delaying modes*.

In general, the *delaying set DS*, i.e. the set of all minimal delaying alternatives, is equal to

$$DS = \left\{ D_d \;\middle|\; \begin{array}{l} D_d \subset S(t^*) \text{ and } \forall \text{ resource } k: \sum_{i \in S(t^*)} r_{ik} - \sum_{i \in D_d} r_{ik} \leq a_k \\ \text{and } \forall\, D_{d'} \in \mathrm{DS} \setminus \{D_d\} : D_{d'} \not\subset D_d \end{array} \right\}$$

The set of minimal delaying modes equals:
$M = \left\{ M_m \;\middle|\; M_m = \{k \prec D_d\}, k \in S(t^*) \setminus D_d, D_d \in DS \right\}$. Activity $k$ is called the *delaying activity*: $k \prec D_d$ implies that $k \prec l$ for all $l \in D_d$.

***THEOREM* 4.** *The delaying strategy which consists of delaying all minimal delaying alternatives $D_d$ by each activity $k \in S(t^*) \setminus D_d$ will lead to the optimal solution in a finite number of steps.*

PROOF. See De Reyck and Herroelen (1998a).

Each minimal delaying mode is then examined for precedence-feasibility and evaluated by computing the critical path-based lower bound $lb_0$. Each precedence-feasible minimal delaying mode with a lower bound $lb_0<T$ is then considered for further branching, which occurs from the node with smallest $lb_0$. If the node represents a project network in which a resource conflict occurs, a new branching occurs. If it represents a feasible schedule, the upper bound $T$ is updated and the procedure backtracks to the previous level in the search tree. Branching occurs until at a certain level in the tree, there are no delaying modes left to branch from. Then, the procedure backtracks to the previous level in the search tree and reconsiders the other delaying modes (not yet branched from) at that level. The procedure stops upon backtracking to level 0.

The fact that semi-active timetabling cannot be applied when dealing with both minimal and maximal time lags results in a different solution strategy employed by the GDH and DRH procedures. In the GDH procedure, *partial* schedules based on the precedence relations are constructed, until a resource conflict is observed. The remainder of the schedule need not be computed. A resource conflict is resolved through the delay of activities participating in the conflict, which corresponds to the addition of precedence relations. The search then advances through time to subsequent resource conflicts until the dummy end activity is scheduled. Upon finding such a complete (precedence- and resource-feasible) schedule, the procedure backtracks, which corresponds to a partial destruction of the schedule. Contrary, in the DRH procedure, in each node of the search tree, *complete* (precedence-based) schedules are evaluated. The first resource conflict in that schedule is subsequently resolved using additional precedence relations. However, contrary to the GDH procedure, the procedure cannot proceed through time, because in a newly derived schedule, obtained by resolving a resource conflict at time $t$, a new conflict can occur at a time instant $t'<t$. When a precedence- and resource-feasible schedule is encountered, the procedure backtracks to the previous level in the search tree which corresponds to removing precedence relations from the project network.

### 4.6.2. Node fathoming rules

Nodes are fathomed if they represent a precedence-infeasible project network or when $lb_0$ exceeds (or equals) $T$. Four additional node fathoming rules (three dominance rules and a new lower bound) and a procedure which reduces the solution space and which can be executed as a pre-processing rule are added. These rules are given below. Additional information and proofs can be found in De Reyck and Herroelen (1998a).

***THEOREM 5.*** *If there exists a minimal delaying alternative $D_d$ with activity $i \in D_d$ but its real successor $j \notin D_d$ ($d_{ij} \geq 0$), we can extend $D_d$ with activity $j$. If the resulting delaying alternative becomes non-minimal as a result of this operation, it may be eliminated from further consideration.*

***THEOREM 6.*** *When a minimal delaying alternative $D_d$ gives rise to two minimal delaying modes $M_{m_1}$ and $M_{m_2}$ with delaying activities $i$ and $j$ respectively, $M_{m_2}$ is dominated by $M_{m_1}$ if $d_{ij} + d_j \geq d_i$. If $d_{ij} + d_j = d_i$ and $d_{ji} + d_i = d_j$, either delaying mode $M_{m_1}$ or $M_{m_2}$ can be dominated.*

Mingozzi et al. (1998) have developed five new lower bounds for the RCPSP, namely $lb_1$, $lb_2$, $lb_p$, $lb_x$ and $lb_3$, derived from different relaxations of a new mathematical formulation. They compute $lb_3$ using a heuristic for the maximum weighted independent set problem. Demeulemeester and Herroelen (1997a) have incorporated another version of $lb_3$ in their GDH algorithm, which can be extended to the RCPSP-GPR as follows. For each activity $i \in V$, its *companions* are determined (the activities with which it can be scheduled in parallel, respecting both the precedence ($d_{ij} < d_i$ and $d_{ji} < d_j$) and resource constraints ($\forall k \leq m: r_{ik} + r_{jk} \leq a_k$)). All activities $i$ are then entered in a list $L$ in non-decreasing order of the number of companions (non-increasing duration as tie-breaker). The following procedure then yields a lower bound, $lb_3^g$. For each activity, we define a remaining duration $d_j^r$. Initially, all $d_j^r$ are equal to $d_j$ and $lb_3^g = 0$.

**while** $L$ not empty **do**
    take the first activity (activity $i$) in $L$ and remove it from $L$
    $lb_3^g = lb_3^g + d_i^r$
    **for** every companion $j$ of $i$ **do**
        **if** $d_{ij} > 0$ **then** $d_j^r = d_j^r - (d_i - d_{ij})$

**else if** $d_{ji} > 0$ **then** $d_j^r = d_j^r - \min\{d_j - d_{ji}, d_i\}$

**else** $d_j^r = d_j^r - d_i$

**endif**

if $d_j^r \leq 0$, remove activity $j$ from $L$

**enddo**

**enddo**

***THEOREM 7.*** *$lb_3^g$ is a valid lower bound for problem* $m,1,va|gpr,\rho_i,\delta_i,vr|C_{\max}$.

$lb_3^g$ is used to fathom nodes for which $lb_3^g \geq T$. However, whereas $lb_0$ is calculated immediately upon the creation of a node, the calculation of $lb_3^g$ is deferred until a decision has been made to actually branch from that node. The rationale behind this is that (a) $lb_3^g$ is more difficult to compute than $lb_0$, and (b) calculating $lb_3^g$ implies calculating the entire matrix $D$. We defer the calculation of $lb_3^g$ and $D$ until the node is actually selected for branching. As a result, only $lb_0$ is used as a branching criterion.

***THEOREM 8.*** *If the set of added precedence constraints which leads to the project network in node x contains as a subset another set of precedence constraints leading to the project network in a previously examined node y in another branch of the search tree, node x can be fathomed.*

***THEOREM 9.*** *If* $\exists i,j \in V$ *and* $k \leq m$ *for which* $r_{ik} + r_{jk} > a_k$ *and* $-d_j < d_{ij} < d_i$, *we can set* $l_{ij} = d_i$.

### *4.6.3. Extensions to other objective functions*

In real-life project scheduling applications, the minimization of the project length is undoubtedly the most popular objective function. Minimizing the project makespan implies that the resources tied up in the activities of the project are released as soon as possible, thereby making them available for other projects in the future. Also, minimizing the project length releases tied-up capital because in many projects, the majority of income payments occur at the end of a project (Kolisch 1996).

Nevertheless, for many actual project scheduling applications, minimizing the project length may be a misrepresentation of the actual

conditions, in which considerations such as cost minimization, tardiness minimization and revenue maximization may be much more relevant. In the literature, a rich variety of objective functions has been the subject of extensive study. These objective functions can be classified into two distinct classes: regular and nonregular measures of performance. A regular measure of performance (which is to be minimized) is a nondecreasing function of the activity completion times. When not imposed by resource, precedence or temporal constraints, it will not be advantageous to delay activities solely to obtain an improved performance under a regular measure of performance. For a nonregular objective function, the condition above does not hold. This implies that delaying activities may improve the performance of the schedule, even if such a delay is not imposed by any constraints.

**4.6.3.1. Regular performance measures**. Practical applications of regular measures of performance often take the form of a cost function based on the activity completion times. Such cost functions may take the following form:

- Minimizing project costs determined by a weighted function of the tardiness of the activities with respect to pre-set due dates $(m,1,va|gpr,\rho_i,\delta_i,vr|\sum w_i T_i)$, where $T_i = \max\{f_i - \overline{\delta_i}, 0\}$, $\overline{\delta_i}$ represents the due date of activity $i$ (not to be confused with the activity deadlines $\delta_i$ which can also be present and which constitute hard constraints that cannot be violated) and $w_i$ denotes the weight (penalty) associated with an additional delay (of one time period) of activity $i$ beyond its due date. We may also want to minimize the number of tardy activities $(m,1,va|gpr,\rho_i,\delta_i,vr|n_T)$ or the maximal activity tardiness $(m,1,va|gpr,\rho_i,\delta_i,vr|T_{max})$.
- Minimize the mean flow time: $m,1,va|gpr,\rho_i,\delta_i,vr|\overline{F}$, where $\overline{F} = \frac{1}{n}\sum_{i=1}^{n}(f_i - \rho_i)$ and $\rho_i$ denotes the release date of activity $i$.

In the DRH procedure, we evaluate the project networks in each node of the search tree by computing an *ESS* (by means of a longest path matrix *D*), which yields a critical path-based lower bound $lb_0$. If we optimize any other *regular* measure of performance $(m,1,va|gpr,\rho_i,\delta_i,vr|reg)$, we can still use the *ESS* to evaluate the project networks and simply replace the calculation of $lb_0$ by the regular performance measure under consideration. The branching strategy based on minimal delaying modes (Theorem 4) can also be used when dealing with other regular performance measures. Also Theorems 5, 6, 8 and 9 are still applicable. Therefore, only two slight modifications are

needed to extend the procedure. First, we need to replace $lb_0$ by the new measure and use the resulting value as a lower bound. Second, the lower bound $lb_3^g$ can no longer be used as a node fathoming rule since it is based on the minimum makespan objective.

**4.6.3.2. Nonregular performance measures.** If we optimize a *nonregular* measure of performance ($m,1,va|gpr,\rho_i,\delta_i,vr|nonreg$), the branching strategy based on minimal delaying modes to resolve resource conflicts (Theorem 4) can still be used. However, we *cannot* use the *ESS* anymore to compute the objective function value and replace the calculation of $lb_0$ by the performance measure under consideration. Rather, the project networks in each node of the search tree should be optimized using the nonregular objective function while discarding the resource constraints ($gpr|nonreg$). Also resource-feasibility should be checked against the schedule obtained by optimizing the nonregular objective function for the resource-unconstrained project network.

A popular nonregular performance measure is the maximization of the net present value (*npv*) of the project, in which positive and negative cash flows are associated with the activities ($m,1,va|gpr,\rho_i,\delta_i,vr,c_i|npv$): Maximize $\sum_{i=1}^{n} c_i e^{-\alpha f_i}$ with $c_i = \sum_{t=1}^{d_i} g_{it}\, e^{\alpha(d_i - t)}$, the cash flow (positive or negative) associated with each activity compounded up to its completion.

For a review of project scheduling problems in which financial considerations are explicitly included, we refer the reader to Herroelen et al. (1997). When maximizing the project *npv*, the evaluation and optimization of the project networks in each node should be accomplished by maximizing the *npv* of the corresponding (precedence-feasible, but not necessarily resource-feasible) project without taking the resource constraints into account ($gpr,\delta_n,c_i|npv$). Algorithms for the unconstrained max-*npv* project scheduling problem ($cpm,\delta_n,c_i|npv$) can be found in Russell (1970), Grinold (1972), Elmaghraby and Herroelen (1990), Herroelen and Gallens (1993) and Herroelen et al. (1996). Unfortunately, none of these algorithms can cope with GPRs ($gpr,\rho_i,\delta_i,c_i|npv$). De Reyck and Herroelen (1996b) have developed an exact recursive enumeration procedure for optimizing the *npv* in project networks with GPRs ($gpr,\rho_i,\delta_i,c_i|npv$), which will be briefly reviewed in the next section.

**4.6.3.3. Maximizing the net present value of projects: the resource-unconstrained case.** The algorithm of De Reyck and Herroelen (1996b) for problem $gpr,\rho_i,\delta_i,c_i|npv$ is based on the intuitive idea that activities carrying

positive cash flows should be executed as early as possible, whereas activities with a negative cash flow should be delayed as much as possible. The procedure consists of 3 steps. In STEP 1, the longest path matrix $D$ is computed. If the project is precedence-feasible, the *early tree* is computed, which spans all activities scheduled at their earliest start time. For every activity $i$, a predecessor $j$ is determined for which $d_{1,j} + d_{j,i} = d_{1,i}$, upon which activities $j$ and $i$ are linked.

The *current tree* is computed in STEP 2 of the algorithm by delaying all activities $i$ with a negative cash flow $c_i$ and no successor in the early tree as much as possible within the early tree, i.e. without affecting the start times of the successor activities in the constraint digraph. This results in a local optimum which cannot be improved by delaying single activities and will reduce the number of recursions required in STEP 3 of the procedure which examines the simultaneous delay of activities. If any activity $i$ has been delayed while computing the current tree, STEP 2 is repeated. After STEP 2 has been repeated a sufficient number of times, the procedure enters a recursive search in STEP 3, in which partial trees $PT$ (with a negative *npv*) are identified that can be disconnected from the current tree and shifted forwards in time in order to increase the *npv* of the project. When such a partial tree is found, the algorithm computes the maximal shift of the partial tree by identifying the maximal possible increase in the start times of the activities belonging to the partial tree without violating any of the precedence constraints, keeping all activities not belonging to $PT$ at their current start times. Therefore, a new arc is determined with minimal displacement, i.e. an arc $(k,l)$ ($k \in PT, l \notin PT$) with minimal value for $d_{1,l} - d_{1,k} - d_{k,l}$. We disconnect the partial tree from the remainder of the current tree and we add the arc $(k,l)$ to the current tree, thereby relinking the forward-shifted partial tree to the current tree. Then, we update the completion times of the activities in the partial tree as follows: $\forall j \in PT$:

$$d_{1,j} = d_{1,j} + \min_{\substack{k \in PT \\ l \notin PT}} \left\{ d_{1,l} - d_{1,k} - d_{k,l} \right\}.$$

If a shift has been found and implemented, the recursive procedure is restarted until no further shift can be accomplished. Then, the optimal schedule with its corresponding *npv* is reported.

**4.6.3.4. Maximizing the net present value of projects: the resource-constrained case.** De Reyck and Herroelen (1998b) have developed a branch-and-bound procedure for problem $m,1,va|gpr,\rho_i,\delta_i,vr,c_i|npv$ based on the DRH algorithm for the minimum makespan case, using the recursive search procedure described above for the calculation of a bound on the project *npv*. Each time a node in the branch-and-bound tree is chosen to

branch from, the corresponding longest path matrix $D$ is computed and the schedule which optimizes the project *npv* is computed, yielding an upper bound on the *npv*. However, in the DRH procedure, when a number of nodes are created at a certain level of the search tree (not yet chosen to be branched from), the matrices $D$ are not yet computed. Therefore, it is not possible to use the algorithm for the unconstrained *npv* maximization to compute an upper bound on the project *npv*. Therefore, the computation of the upper bound *ub* on the project *npv* is not made upon creation of a node, but is deferred until a decision has been made to actually branch from it. As a result, another criterion (a myopic criterion based on the cash flows of the delayed activities) is used in order to select the node to branch from at a certain level.

Another often encountered example of a nonregular performance measure is the minimization of the weighted earliness-tardiness of the activities in a project, in which a due date, earliness penalties as well as tardiness penalties are associated with the activities ($m,1,va|gpr,\rho_i,\delta_i,vr|early/tardy$): $\sum_{i=1}^{n} w_i \sum_{t=es_i}^{ls_i} \left| t + d_i - \overline{\delta_i} \right| x_{it}$ . In that case, the project network in each node of the search tree should be optimized such that a minimum penalty value due to earliness or tardiness of the activities is obtained, while the activities are subject to GPRs only ($gpr,\rho_i,\delta_i|early/tardy$). Exact solution procedures for optimizing due date performance in project networks are sparse. To the best of our knowledge, if the precedence relations among the activities are allowed to be GPRs, no solution procedure is available at all for minimizing earliness-tardiness-based objective functions. This constitutes a promising area for future research.

**4.6.3.5. Multiple objective functions.** From the discussion above, it is clear that project management has the choice between a wide variety of performance measures. These measures may pertain to the makespan of the project, the tardiness of activities or subprojects, the activity flow times, the levelness of the resource profile(s) and may even include financial considerations. In many situations, several of these objective functions may be relevant at the same time. Often, however, the relevant objectives are in conflict. In that case, a trade-off will be present in the sense that the project manager will have to decide which performance measure is the most important, in which order they should be considered or which weights should be assigned to each of the measures.

This gives rise to the problem of scheduling projects under multiple objectives. We distinguish between the case where (a) multiple objectives are considered in a pre-specified order or have been assigned a weight

determining the trade-off between the measures, and (b) where the solution method should present a series of alternative solutions from which the decision maker should select a solution based on his/her perspectives of the relation between the performance measures. In the former case, the solution procedure can unambiguously determine the optimal solution because the multiple objectives can be merged into a single objective function. In the latter case, the solution procedure cannot determine an optimal schedule because a trade-off between the various performance measures has not been firmly established. The procedure should then present a number of efficient solutions, from which the decision maker can select a schedule.

If multiple regular performance measures are considered, each one given a weight to determine its importance vis-à-vis the other measures (or a rank order), the DRH procedure can still be used. In that case, the *ESS* can still be used to evaluate the resource-unconstrained project networks in each node of the search tree. If, however, nonregular performance measures are considered, the problem becomes much more complex. In that case, the resource-unconstrained projects should be optimized taking into account the weighted nonregular (and regular) objective functions. Also when no weights or strict order can be assigned to the measures, the solution approach should be modified rather extensively. In that case, the procedure should present multiple viable alternatives and allow the user to determine which schedule he/she prefers based on the associated values for the various objective functions.

## 4.7. Modifications to the original algorithms

The GDH and DRH procedures have been recoded and compiled using Microsoft Visual C++ 4.0 under Windows NT for use on a Dell Pentium Pro-200Mhz personal computer. The GDH code requires 91 Kb, whereas the data structures are allowed to use up to 16 Mb. This memory is mainly allocated to the application of the cutset dominance rule. For the DRH code, which requires 90 Kb, only 400 Kb should be reserved for storing the data.

The GDH procedure has undergone some modifications since its development. First, we have corrected the application of the cutset dominance rule as explained in Section 4.5.2. Secondly, we now only consider efficient cutsets when applying the cutset dominance rule: every time a new cutset is saved, all cutsets that are dominated by it are removed, resulting in a significantly smaller set of cutsets and a substantial speed-up of the dominance rule. The codes of both algorithms have been modified in order to take full advantage of modern 32-bit compiler architecture. This results in a significant efficiency gain. The major change in the GDH

procedure involves a new coding scheme for the cutset dominance rule. Similar adjustments have been described for the case with zero-lag finish-start precedence relations (see Demeulemeester and Herroelen 1997a). Other changes involve merging different resource types into one global resource type (using 32-bit integers). Additional code polishing also leads to an increase in performance. For technical details we refer the reader to Demeulemeester and Herroelen (1997a).

## 4.8. Computational experience

### *4.8.1. Previous computational experience*

**4.8.1.1. The GRCPSP ($m,1,va|min,\rho_i,\delta_i|C_{max}$).** Computational experience with the GDH procedure is reported by Demeulemeester and Herroelen (1997b) on the problem set consisting of the 110 RCPSP instances assembled by Patterson (1984). The results were very promising and indicated that the algorithm was, on the average, only 2.5 times slower than the similar procedure designed for the RCPSP. For the Simpson problem set, consisting of the same 110 RCPSP instances, extended with variable resource availabilities by Simpson et al. (1992), the procedure also performed substantially better than the procedure of Simpson et al. (1992). A third problem set consisted of ten problem instances based on Patterson problem 72 in which ready times, deadlines and precedence diagramming constraints were introduced. All ten instances could be solved very quickly (in at most 32.08 seconds).

The changes in the coding of the GDH procedure result in a dramatic decrease in the computation times for these problem instances. Using a Pentium Pro-200Mhz computer, the average computation time for the Simpson problem set decreases by a factor of more than one thousand (0.009 seconds versus 9.273 seconds). For the ten problems based on Patterson problem 72, the speed-up factor is only 12 (0.434 seconds versus 0.035 seconds).

**4.8.1.2. The RCPSP-GPR ($m,1,va|gpr,\rho_i,\delta_i,vr|C_{max}$).** De Reyck and Herroelen (1998a) report computational results on three different problem sets in order to validate the DRH procedure against the serial and parallel heuristics developed by Franck and Neumann (1996). These heuristics improve upon the procedures developed by Neumann and Zhan (1995), Zhan (1994) and Brinkmann and Neumann (1996). All three data sets have been generated using the random problem generator ProGen/max developed by Schwindt (1996). The first problem set consists of 1,080 100-activity

instances. The second set consists of 1,440 100-activity problem instances. The third set consists of 7,200 instances with 10 up to 100 activities.

Table 4:1 shows some computational results on the first problem set. These results are obtained using Microsoft Visual C++ 2.0 under Windows NT for a Digital Venturis Pentium-60 personal computer with 16Mb of internal memory. The branch-and-bound procedure is truncated after a specific amount of running time (1, 10 and 100 seconds). The results include the number of problems solved to optimality (for which the optimum was found *and verified*), the number of problems for which the optimal solution is obtained (but *not necessarily verified*), the number of problems for which the best known solution is obtained, the number of unsolved problems (for which a feasible solution could not be determined and neither infeasibility of the instance could be proven), the average deviation from a lower bound and the average deviation from the best known solution. The lower bound $lb$ used to compute the deviations, is the maximum of the critical path-based lower bound $lb_0$, the resource-based lower bound $lb_r = \max_{k=1}^{m}\left\{\left\lceil \sum_{i=1}^{n} d_i r_{ik} / a_k \right\rceil\right\}$ and $lb_3^g$ (computed in the root node of the search tree after pre-processing). The column labelled F&N in Table 4:1 contains the results obtained by Franck and Neumann (1996), which are obtained by running a collection of 44 different heuristics which rank among the best currently available. The best known solution referred to in Table 4:1 is the best of the solutions obtained with various versions of the DRH algorithm running for up to 1 hour per problem and with the heuristic (F&N) solutions, and can therefore be considered as near-optimal.

**Table 4:1.** The results on problem set I

| | F&N | DRH 1 sec | DRH 10 sec | DRH 100 sec |
|---|---|---|---|---|
| Solved to optimality | 196 (18%) | 543 (50%) | 592 (55%) | 609 (56%) |
| Optimal solution is found | 220 (20%) | 578 (54%) | 596 (55%) | 609 (56%) |
| Best known solution is found | 378 (35%) | 606 (56%) | 652 (60%) | 682 (63%) |
| Unsolved problems | 21 (2%) | 205 (19%) | 86 (8%) | 68 (6%) |
| Average deviation from *lb* | 17.02% | 5.99% | 9.77% | 10.00% |
| Average deviation from best solution | 7.20% | 2.20% | 2.54% | 2.31% |

The DRH procedure manages to solve more than 50% of the 100-activity problem instances to optimality within 1 second of computation time. However, increasing the allowed computation time from 1 second to 100

seconds leads to an increase of only 12% in the problem instances solved to optimality (from 543 to 609). The average deviation from the best known solution (lower bound) never exceeds 2.54% (10.00%), whereas the F&N heuristics result in an average deviation of 7.20% (17.02%). Less reassuring, however, is that, especially for small time limits, a relatively large number of problems remains unsolved. The F&N heuristics do a better job on this issue.

This inspired us to develop another approach which is based on finding a feasible solution first, rather than going immediately for the optimal solution. When no feasible solution has been obtained yet, this approach uses a new criterion (referred to as time window slack *TWS*) to decide on which node to branch from, based on feasibility criteria. The node that entails the highest chance of leading to a feasible solution is selected first, regardless of its lower bound (which is only used as a tie-breaker). When a feasible solution is obtained, again the lower bound is used as a branching criterion. Using this new approach, the number of unsolved problems decreases to 27 (2.5%), 8 (0.7%) and 6 (0.6%) for the three time limit settings, whereas the number of problems solved to optimality does not significantly differ from the original approach. The average deviation from the best known solution (lower bound) increases somewhat, but never exceeds 4.5% (14%), thereby still outperforming the heuristics. More details can be found in De Reyck and Herroelen (1998a).

### *4.8.2. New computational results*

**4.8.2.1. Benchmark problem set.** In this section, we present new computational experience with the two enhanced branch-and-bound procedures on a new benchmark problem set consisting of 1,620 randomly generated instances. The parameters used to generate the new problem set are given in Table 4:2. The indication [*x*,*y*] means that the corresponding value is randomly generated in the interval [*x*,*y*], whereas *x*; *y*; *z* means that three settings for that parameter were used in a full factorial experiment. For each combination of parameter values, 10 instances have been generated.

The resource factor *RF* (Pascoe 1966) reflects the average portion of resources requested per activity. If *RF*=1, then each activity requests all resources. *RF*=0 indicates that no activity requests any resource: $RF = \frac{1}{nK}\sum_{i=1}^{n}\sum_{k=1}^{K}\begin{cases}1, \text{ if } r_{ik} > 0\\ 0, \text{ otherwise}\end{cases}$. The *resource strength RS* (Cooper 1976) is redefined by Kolisch et al. (1995) as $\left(a_k - r_k^{\min}\right)/\left(r_k^{\max} - r_k^{\min}\right)$, where $a_k$ is

the total availability of renewable resource type $k$, $r_k^{\min} = \max_{i=1,\ldots,n} r_{ik}$ (the maximum resource requirement for each resource type), and $r_k^{\max}$ is the peak demand for resource type $k$ in the precedence-based early start schedule. The resource availability is assumed to be constant over time.

**Table 4:2.** The parameter settings of the new problem sets

| Control parameter | Values |
|---|---|
| number of activities | 10; 20; 30 |
| activity durations | [2,10] |
| number of resource types | 4 |
| minimum / maximum number of resources used per activity | 1 / 4 |
| activity resource demand | [1,10] |
| resource factor, *RF* (Pascoe 1966) | 0.50; 1.00 |
| resource strength, *RS* (Kolisch et al. 1995) | 0.00; 0.25; 0.50 |
| number of initial and terminal activities | [2,4] |
| maximum number of predecessors and successors | 3 |
| order strength, *OS* (Mastor 1970) | 0.35; 0.50; 0.65 |
| % maximal time lags | 0%; 10%; 20% |
| number of cycle structures (Brinkmann and Neumann 1996) | [1,10] |
| minimum / maximum number of nodes per cycle structure | 2 / 30 |
| coefficient of cycle structure density (Schwindt 1996) | 0.3 |
| cycle structure tightness (Schwindt 1996) | 0.5 |

The *order strength OS* is defined as the number of precedence relations, including the transitive ones, divided by the theoretical maximum of such precedence relations, namely $n(n\text{-}1)/2$, where $n$ denotes the number of activities (Mastor 1970). Because *OS* only applies to acyclic networks, it is applied to the acyclic skeleton of the generated project networks obtained by ignoring all maximal time lags (for details see Schwindt 1996). For the definition of cycle structures and related measures, we refer the reader to Schwindt (1996).

**4.8.2.2. Overall results.** The problems without any maximal time lags correspond to instances of the GRCPSP, whereas the problems with 10% and 20% maximal time lags correspond to instances of the RCPSP-GPR. Therefore, we solved the former (540) instances with the GDH procedure,

and the latter (1,080) instances with the DRH procedure. A time limit of 1,000 seconds is imposed. The overall results can be found in Table 4:3.

**Table 4:3.** Overall results

| | 0% | | 10% | | 20% | | All problems | |
|---|---|---|---|---|---|---|---|---|
| *n* | Optimal | CPU-time | Optimal | CPU-time | Optimal | CPU-time | Optimal | CPU-time |
| 10 | 180 | 0.001 | 180 | 0.00 | 180 | 0.00 | 540 | 0.00 |
| 20 | 180 | 0.005 | 180 | 0.11 | 180 | 0.12 | 540 | 0.08 |
| 30 | 180 | 0.021 | 177 | 36.37 | 178 | 34.27 | 535 | 23.56 |
| All | 540 | 0.009 | 537 | 12.16 | 538 | 11.46 | 1,615 | 7.88 |

The GRCPSP instances can be solved to optimality within very small CPU-times using the GDH procedure. However, the RCPSP-GPR instances, solved using the DRH procedure, require much more time. Five out of the 1,080 instances cannot be solved to (verified) optimality within 1,000 seconds. This illustrates the much higher complexity of the RCPSP-GPR versus the GRCPSP. The main reason for the difference in complexity is that for solving the RCPSP-GPR, semi-active timetabling cannot be applied. Consequently, most of the concepts developed for the minimal time lag case are not transferable to the GPR-case. The fundamental logic of the GDH procedure, namely its branching strategy, is based on the principle of semi-active timetabling and can therefore not be applied for the GPR case. Also the dominance rules, including the left-shift dominance rule and the powerful cutset dominance rule, which is mainly responsible for the efficiency of the GDH procedure, are not applicable anymore. Consequently, the DRH procedure is based on a different branching strategy and a new set of dominance rules and lower bounds.

The fact that the branching strategy and the dominance rules which are applicable for the GPR case are less powerful than those used for the precedence diagramming case, can be illustrated by comparing the performance of both algorithms on the problem instances with minimal time lags only. Whereas the GDH procedures solves all 540 instances to optimality with an average CPU-time of 0.009 seconds, the more general DRH procedure cannot solve 12 out of the 540 instances within 1,000 seconds. Naturally, when no maximal time lags are present, the DRH procedure is no longer efficient because it is designed for the inclusion of maximal time lags. In that case, the GDH procedure should be used. Similarly, when all precedence relations are of the zero-lag finish-start type,

the procedure of Demeulemeester and Herroelen (1992, 1997a) should be used instead. The efficiency of the DRH procedure heavily depends on the relative number of maximal time lags in the problem instances. The more maximal time lags, the more effective the dominance and bounding rules, and the more efficient the DRH procedure.

**4.8.2.3. The modified cutset dominance rule.** We examined the impact of the new cutset dominance rule on the efficiency of the GDH procedure by implementing the original cutset dominance rule in the new procedure. The results indicate that the efficiency of the procedure does not substantially differ. For all but 1 instance, the optimal solution is obtained despite the use of the erroneous cutset dominance rule. The computation times using the modified cutset dominance rule are only slightly higher (0.009 versus 0.008 seconds).

**4.8.2.4. Results with truncated procedures.** Table 4:4 presents the results with a truncated version of the DRH procedure. We did not report any results with a truncated GDH procedure since it is able to solve all of the instances with minimal time lags to optimality with very little computational effort. Therefore, we report the results with the DRH procedure only (also for the instances without maximal time lags). The results reported are the number of instances for which the optimal solution is found (not necessarily verified, including the problems proven to be infeasible) and the average deviation from the best known solution (all but five of these solutions are known to be optimal). The deviations are only computed for the instances which are feasible and for which the truncated procedure was able to find a feasible solution.

Although the DRH procedure cannot solve all instances to optimality when the imposed time limit is rather small, the obtained heuristic solutions are of high quality, especially when the relative amount of maximal time lags is rather high. The results conform to the results of previous computational experiments (De Reyck and Herroelen 1998a), which show a similar performance of the truncated DRH procedure on instances with up to 100 activities (see also Table 4:1).

**4.8.2.5. Impact of problem characteristics.** In Table 4:5, the impact of the order strength *OS* on the complexity of the problem instances is examined. Clearly, *OS* has a negative impact on the problem complexity, measured by the number of problems solved to optimality and the required CPU-time. This result is in line with other results reported in the literature (De Reyck 1995, Schwindt 1996, De Reyck and Herroelen 1998a).

**Table 4:4.** Heuristic results

| | | 0% | | 10% | | 20% | | All problems | |
|---|---|---|---|---|---|---|---|---|---|
| | | Optimal | % dev. | Optimal | % dev. | Optimal | % dev. | Optimal | % dev. |
| time limit 1 SECOND | $n = 10$ | 180 | 0.00% | 180 | 0.00% | 180 | 0.00% | 540 | 0.00% |
| | $n = 20$ | 168 | 0.26% | 178 | 0.06% | 179 | 0.01% | 525 | 0.11% |
| | $n = 30$ | 112 | 2.23% | 152 | 1.23% | 148 | 1.68% | 412 | 1.71% |
| | All | 460 | 0.83% | 510 | 0.43% | 507 | 0.56% | 1,477 | 0.61% |
| time limit 10 SECONDS | $n = 10$ | 180 | 0.00% | 180 | 0.00% | 180 | 0.00% | 540 | 0.00% |
| | $n = 20$ | 177 | 0.05% | 180 | 0.00% | 180 | 0.00% | 537 | 0.02% |
| | $n = 30$ | 131 | 1.31% | 167 | 0.59% | 164 | 0.76% | 462 | 0.89% |
| | All | 488 | 0.46% | 527 | 0.20% | 524 | 0.25% | 1,539 | 0.30% |
| time limit 100 SECONDS | $n = 10$ | 180 | 0.00% | 180 | 0.00% | 180 | 0.00% | 540 | 0.00% |
| | $n = 20$ | 180 | 0.00% | 180 | 0.00% | 180 | 0.00% | 540 | 0.00% |
| | $n = 30$ | 151 | 0.65% | 172 | 0.30% | 173 | 0.14% | 496 | 0.36% |
| | All | 511 | 0.22% | 532 | 0.10% | 533 | 0.05% | 1,576 | 0.12% |

**Table 4:5.** Impact of *OS*

| | $OS = 0.35$ | | $OS = 0.50$ | | $OS = 0.75$ | |
|---|---|---|---|---|---|---|
| | Optimal | CPU-time | Optimal | CPU-time | Optimal | CPU-time |
| $n = 10$ | 180 | 0.00 | 180 | 0.00 | 180 | 0.00 |
| $n = 20$ | 180 | 0.89 | 180 | 0.31 | 180 | 0.06 |
| $n = 30$ | 166 | 136.22 | 175 | 58.29 | 177 | 22.38 |
| All problems | 526 | 45.71 | 535 | 19.53 | 537 | 7.48 |

The impact of the resource-based measures *RF* and *RS* is given in Tables 4:6 and 4:7. *RF* has a strong impact on the complexity of the problem. The higher *RF*, the harder the corresponding problem instances. These results conform to the conclusions drawn by other research for related problem types (Kolisch et al. 1995, De Reyck and Herroelen 1998a). The effect of *RS* is not monotonously increasing or decreasing. On the contrary, it is bell-shaped, the hardest instances corresponding to an intermediate *RS* setting (equal to 0.25). However, there is a clear difference between the complexity of problems with small or large *RS* values. Problems with small *RS* values (*RS*=0) are much more difficult than problems with a high *RS* value (*RS*=0.5). Therefore, the 'top' of the bell-shaped complexity curve is

skewed towards *RS*=0. The impact of *RS* corresponds to the conjecture of Elmaghraby and Herroelen (1980) and the results of De Reyck and Herroelen (1996a) for the RCPSP.

**Table 4:6.** Impact of *RF*

| | *RF* = 0.50 | | *RF* = 1.00 | |
|---|---|---|---|---|
| | Optimal | CPU-time | Optimal | CPU-time |
| *n* = 10 | 270 | 0.00 | 270 | 0.78 |
| *n* = 20 | 270 | 0.00 | 269 | 14.66 |
| *n* = 30 | 270 | 0.06 | 249 | 129.94 |
| All problems | 810 | 0.02 | 788 | 48.46 |

**Table 4:7.** Impact of *RS*

| | *RS* = 0.00 | | *RS* = 0.25 | | *RS* = 0.50 | |
|---|---|---|---|---|---|---|
| | Optimal | CPU-time | Optimal | CPU-time | Optimal | CPU-time |
| *n* = 10 | 180 | 0.00 | 180 | 0.00 | 180 | 0.00 |
| *n* = 20 | 180 | 0.98 | 180 | 0.21 | 180 | 0.07 |
| *n* = 30 | 173 | 93.63 | 166 | 112.35 | 179 | 10.91 |
| All problems | 533 | 31.54 | 526 | 37.52 | 539 | 3.66 |

**4.8.2.6. Variable resource availabilities.** When the resource availabilities are allowed to vary over time, the complexity of the GRCPSP and the RCPSP-GPR increases. The GDH procedure needs to be explicitly equipped with the ability to handle such time-varying resource availabilities, which will result in an increased number of decision periods and nodes in the search tree. The DRH procedure need not be modified in order to be able to handle time-varying resource availabilities (or, for that matter, variable resource requirements). The introduction of dummy activities and appropriate time lags, as discussed in Section 4.4, will transform an instance with variable availabilities (and requirements) into an equivalent instance with constant availabilities (and requirements). Naturally, the increased number of activities in the project network will have a substantial effect on the efficiency of the DRH procedure.

In order to estimate the effect of introducing variable resource availabilities on the performance of the GDH procedure, we modified the

540 instances with minimal time lags as follows. The constant availabilities are replaced by variable availabilities which are constant for an interval equal to 5 time periods. The availability is varied from interval to interval by increasing, respectively decreasing the availability with 1 or 2 units (or by keeping it constant), each with equal probability. Each time the resource availability dropped below the maximal demand of any of the activities for that resource type, the availability was assigned that maximal demand. The computational results indicate that the performance of the GDH procedure does not suffer significantly from this relaxed assumption. The average computation time increases from 0.009 to 0.018 seconds, while the average number of nodes in the search tree increases from 781 to 1,492.

## 4.9. Conclusions

In this chapter, we review a number of algorithms for project scheduling problems with resource constraints and generalized precedence relations. These generalized precedence relations specify minimal and/or maximal time lags between the starting and completion times of activities, and allow to model various types of activity overlaps (either permissible or mandatory), and also allow to model a wide variety of characteristics of real-life project scheduling applications. Also several objective functions are dealt with, including all kinds of regular performance measures and the nonregular measure of maximizing the net present value of a project.

The algorithms are enhanced and recoded in order to gain computational efficiency taking full advantage of modern 32-bit compiler architecture. We report new computational results using these algorithms on a problem set consisting of randomly generated problem instances. A comparison with results reported in the literature reveals that the algorithms presented here constitute the state-of-the-art for project scheduling with generalized precedence relations. When the optimal solution cannot be guaranteed, a truncated version of the algorithms can be used to provide high-quality solutions at acceptable computational cost. The experiments also highlight the fundamental difference in complexity between the precedence diagramming case, i.e. the case with minimal time lags only, and the generalized precedence relations case, in which both minimal and maximal time lags are allowed.

An investigation into the relationship between the complexity of a problem instance, defined by the computational effort required for its solution, and its intrinsic characteristics, reveals that the network morphology as well as the resource-constrainedness of the problems significantly influence the required computational effort. The more dense

the project network becomes, measured by an increase in the *order strength*, the easier it is to obtain the optimal solution. When more activities require the use of resources, measured by an increase in the *resource factor*, the harder the instances become. The resource-constrainedness, measured by the *resource strength*, has a bell-shaped impact on the computational complexity. Instances with a low or high resource-constrainedness are easier to solve than instances with an intermediate resource-constrainedness, although the most difficult problems are relatively highly resource-constrained.

## Acknowledgements

We would like to thank Klaus Neumann and Christoph Schwindt from the University of Karlsruhe for providing us with the project generator ProGen/max and Jan Weglarz for providing the opportunity for contributing to the *Handbook on Recent Advances in Project Scheduling*.

## References

BARTUSCH, M., R. H. MÖHRING, AND F. J. RADERMACHER. 1988. Scheduling Project Networks with Resource Constraints and Time Windows. *Annals O.R.* **16**, 201-240.

BRINKMANN, K. AND K. NEUMANN. 1996. Heuristic Procedures for Resource-Constrained Project Scheduling with Minimal and Maximal Time Lags: The Minimum Project-Duration and Resource-Levelling Problem. *J. Dec. Syst.* **5**, 129-156.

COOPER, D. F. 1976. Heuristics for Scheduling Resource-Constrained Projects: An Experimental Comparison. *Mgmt. Sci.* **22**, 1186-1194.

CRANDALL, K. C. 1973. Project Planning with Precedence Lead / Lag Factors. *Project Mgmt. Quart.* **4**, 18-27.

DE REYCK, B. 1995. On the Use of the Restrictiveness as a Measure of Complexity for Resource-Constrained Project Scheduling. Research Report 9535, Department of Applied Economics, Katholieke Universiteit Leuven.

DE REYCK, B. 1998. *Scheduling Projects with Generalized Precedence Relations - Exact and Heuristic Procedures*. Ph.D. Dissertation, Department of Applied Economics, Katholieke Universiteit Leuven.

DE REYCK, B. AND W. HERROELEN. 1996a. On the Use of the Complexity Index as a Measure of Complexity in Activity Networks, *Eur. J. Opnl. Res.* **91**, 347-366.

DE REYCK, B. AND W. HERROELEN. 1996b. An Optimal Procedure for the Unconstrained Max-*npv* Project Scheduling Problem with Generalized Precedence Relations. Research Report 9642, Department of Applied Economics, Katholieke Universiteit Leuven.

DE REYCK, B. AND W. HERROELEN. 1997. The Multi-Mode Resource-Constrained Project Scheduling Problem with Generalized Precedence Relations. Research

Report 9725, Department of Applied Economics, Katholieke Universiteit Leuven.

DE REYCK, B. AND W. HERROELEN. 1998a. A Branch-and-Bound Algorithm for the Resource-Constrained Project Scheduling Problem with Generalized Precedence Relations, *Eur. J. Opnl. Res.*, to appear.

DE REYCK, B. AND W. HERROELEN. 1998b. An Optimal Procedure for the Resource-Constrained Project Scheduling Problem with Discounted Cash Flows and Generalized Precedence Relations, *Comput. and O.R.* **25**, 1-17.

DEMEULEMEESTER, E. AND W. HERROELEN. 1992. A Branch-and-Bound Procedure for the Multiple Resource-Constrained Project Scheduling Problem. *Mgmt. Sci.* **38**, 1803-1818.

DEMEULEMEESTER, E. AND W. HERROELEN. 1997a. New Benchmark Results for the Resource-Constrained Project Scheduling Problem. *Mgmt. Sci.* **43**, 1485-1492.

DEMEULEMEESTER, E. AND W. HERROELEN. 1997b. A Branch-and-Bound Procedure for the Generalized Resource-Constrained Project Scheduling Problem. *Opns. Res.* **45**, 201-212.

ELMAGHRABY, S. E. 1977. *Activity Networks - Project Planning and Control by Network Models*. Wiley Interscience, New York.

ELMAGHRABY, S. E. AND W. HERROELEN. 1980. On the Measurement of Complexity in Activity Networks, *Eur. J. Opnl. Res.* **5**, 223-234.

ELMAGHRABY, S. E. AND W. HERROELEN. 1990. The Scheduling of Activities to Maximize the Net Present Value of Projects. *Eur. J. Opnl. Res.* **49**, 35-49.

ELMAGHRABY, S. E. AND J. KAMBUROWSKI. 1992. The Analysis of Activity Networks under Generalized Precedence Relations. *Mgmt. Sci.* **38**, 1245-1263.

FRANCK, B. AND K. NEUMANN. 1996. Priority-Rule Methods for the Resource-Constrained Project Scheduling Problem with Minimal and Maximal Time Lags - An Emprical Analysis. *Proceedings of the Fifth International Workshop on Project Management and Scheduling*, 11 - 13 April, Poznan, 88-91.

GRINOLD, R. C. 1972. The Payment Scheduling Problem, *Nav. Res. Log. Quart.* **19**, 123-136.

HERROELEN, W. AND E. GALLENS. 1993. Computational Experience with an Optimal Procedure for the Scheduling of Activities to Maximize the Net Present Value of Projects. *Eur. J. Opnl. Res.* **65**, 274-277.

HERROELEN, W., E. DEMEULEMEESTER, AND P. VAN DOMMELEN. 1996. An Optimal Recursive Search Procedure for the Deterministic Unconstrained Max-npv Project Scheduling Problem. Research Report 9603, Department of Applied Economics, Katholieke Universiteit Leuven.

HERROELEN, W., P. VAN DOMMELEN AND E. DEMEULEMEESTER. 1997. Project Network Models with Discounted Cash Flows: A Guided Tour through Recent Developments, *Eur. J. Opnl. Res.* **100**, 97-121.

HERROELEN, W., E. DEMEULEMEESTER AND B. DE REYCK. 1998. A Classification Scheme for Project Scheduling Problems, in: Weglarz J. (Ed.), *Handbook on Recent advances in Project Scheduling*, Kluwer Academic Publishers, this volume.

KERBOSH, J. A. G. M. AND H. J. SCHELL. 1975. Network Planning by the Extended METRA Potential Method. Report KS-1.1, University of Technology Eindhoven, Department of Industrial Engineering.

KOLISCH, R. 1996. Efficient Priority Rules for the Resource-Constrained Project Scheduling Problem. *J. Opns. Mgmt.* **14**, 179-192.

KOLISCH, R., A. SPRECHER AND A. DREXL. 1995. Characterization and Generation of a General Class of Resource-Constrained Project Scheduling Problems. *Mgmt. Sci.* **41**, 1693-1703.

LAWLER, E. L. 1976. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart and Winston, New York.

MASTOR, A. A. 1970. An Experimental and Comparative Evaluation of Production Line Balancing Techniques. *Mgmt. Sci.* **16**, 728-746.

MINGOZZI, A., V. MANIEZZO, S. RICCIARDELLI, AND L. BIANCO. 1998. An Exact Algorithm for Project Scheduling with Resource Constraints Based on a New Mathematical Programming Formulation. *Mgmt. Sci.*, to appear.

MODER, J. J., C. R. PHILLIPS, AND E. W. DAVIS. 1983. *Project Management with CPM, PERT and Precedence Diagramming*. Van Nostrand Reinhold Company, Third Edition.

NEUMANN, K. AND C. SCHWINDT. 1997. Activity-on-Node Networks with Minimal and Maximal Time Lags and Their Application to Make-to-Order Production. *OR Spektrum*. **19**, 205-217.

NEUMANN, K. AND J. ZHAN. 1995. Heuristics for the Minimum Project-Duration Problem with Minimal and Maximal Time Lags under Fixed Resource Constraints. *J. Intelligent Manufacturing*. **6**, 145-154.

PASCOE, T. L. 1966. Allocation of Resources - CPM, *Rev.fr. Rech. Opér.* **38**, 31-38.

PATTERSON, J. H. 1984. A Comparison of Exact Procedures for Solving the Multiple Resource-Constrained Project Scheduling Problem. *Mgmt. Sci.* **30**, 854-867.

ROY, B. 1962. Graphes et Ordonnancement. *Rev. fr. Rech. Opér.* **25**, 323-333.

RUSSELL, A. H. 1970. Cash Flows in Networks. *Mgmt. Sci.* **16**, 357-373.

SCHWINDT, C. 1996. Generation of Resource-Constrained Project Scheduling Problems with Minimal and Maximal Time Lags. Report WIOR-489, Institut für Wirtschaftstheorie und Operations Research, Universität Karlsruhe.

SCHWINDT, C. AND K. NEUMANN. 1996. A New Branch-and-Bound-Based Heuristic for Resource-Constrained Project Scheduling with Minimal and Maximal Time Lags. *Proceedings of the Fifth International Workshop on Project Management and Scheduling*, 11 - 13 April, Poznan, 212-215.

SIMPSON III, W. P. AND J. H. PATTERSON. 1992. A Parallel Exact Solution Procedure for the Multiple Resource-Constrained Project Scheduling Problem. Research Report, Indiana University.

SPRECHER, A. AND A. DREXL. 1996. Minimal Delaying Alternatives and Semi-Active Timetabling in Resource-Constrained Project Scheduling. Research Report 426, Christian-Albrechts-Universität zu Kiel.

WIEST, J. D. 1981. Precedence Diagramming Methods: Some Unusual Characteristics and their Implications for Project Managers, *J. Opns. Mgmt.* **1**, 121-130.

ZHAN, J. 1994. Heuristics for Scheduling Resource-Constrained Projects in MPM Networks. *Eur. J. Opnl. Res.* **76**, 192-205.