12-1999

The multi– mode resource– constrained project scheduling problem with generalized precedence relations

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

Willy HERROELEN

Citation

DE REYCK, Bert and HERROELEN, Willy. The multi– mode resource– constrained project scheduling problem with generalized precedence relations. (1999). *European Journal of Operational Research*. 119, (2), 538-556.
Available at: https://ink.library.smu.edu.sg/lkcsb_research/6745

ELSEVIER

European Journal of Operational Research 119 (1999) 538–556

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/orms

The multi-mode resource-constrained project scheduling problem with generalized precedence relations

Bert De Reyck [a,*], Willy Herroelen [b]

[a] *London Business School, Sussex Place, Regent's Park, London NW1 4SA, UK*
[b] *Department of Applied Economics, Katholieke Universiteit Leuven, Naamsestraat 69, B-3000 Leuven, Belgium*

Abstract

In this paper, we tackle the challenging problem of scheduling activities to minimize the project duration, in which the activities (a) are subject to generalized precedence relations, (b) require units of multiple renewable, non-renewable and doubly constrained resources for which a limited availability is imposed, and (c) can be performed in one of several different ways, reflected in multiple activity scenarios or modes. These multiple modes give rise to several kinds of trade-offs (time/resource, time/cost and resource/resource trade-offs) which allow for a more efficient allocation and use of resources. We present a local search-based solution methodology which is able to handle many real-life project scheduling characteristics such as time-varying resource requirements and availabilities, activity ready times, due dates and deadlines, activity overlaps, activity start time constraints and other types of temporal constraints.

Keywords: Project management; Planning and scheduling; Generalized precedence relations; Multiple activity modes; Heuristics; Local search; Tabu search

1. Introduction

In this paper, we present a heuristic solution methodology for the *multi-mode resource-constrained project scheduling problem with generalized precedence relations* (*MRCPSP-GPR*). The objective of the MRCPSP-GPR is to schedule a number of activities, subject to generalized precedence relations (minimal and maximal time lags between the activity starting and completion times), which require a specific amount of possibly several renewable, non-renewable and doubly constrained resources. The activities have multiple execution scenarios (reflecting different ways of performing them), each scenario possibly having a different impact on the activity's duration, the costs associated with it and its resource requirements. Multiple activity modes give rise to several types of trade-offs between (a) the activity duration and its use of resources (time/resource trade-off), (b) the activity duration and its cost (time/cost trade-off), and (c) the quantity and combination of resources employed by the activity (resource/resource trade-off).

Table 1 provides an overview of some of the most important related problem types encountered in the project scheduling literature. The problems

* Corresponding author. E-mail: breyck@lbs.ac.uk

0377-2217/99/$ – see front matter
PII: S0377-2217(99)00151-4

Table 1
A classification of project scheduling problems

	Single mode		Multiple mode		
	No resource types	Multiple renewable resource types	1 non-renewable resource type	1 renewable resource type	Multiple renewable and non-renewable resource types
	no trade-offs	no trade-offs	time/cost trade-offs	time/resource trade-offs	time/cost trade-offs time/resource trade-offs resource/resource trade-offs
ZERO-LAG FS	CPM/PERT $cpm\|C_{\max}$	RCPSP $m,1\|cpm\|C_{\max}$	DTCTP $1,T\|cpm,disc,mu\|C_{\max}$	DTRTP $1,1\|cpm,disc,mu\|C_{\max}$	MRCPSP $m,1T\|cpm,disc,mu\|C_{\max}$
MIN SS, SF, FS, FF	PDM $min\|C_{\max}$	GRCPSP $m,1,va\|min,\rho_i,\delta_i\|C_{\max}$	GDTCTP 1, $T\|min,\rho_i,disc,mu\|C_{\max}$	GDTRTP 1, $1\|min,\rho_i,disc,mu\|C_{\max}$	GMRCPSP $m,1T\|min,\rho_i,disc,mu\|C_{\max}$
MIN + MAX SS, SF, FS, FF	MPM $gpr\|C_{\max}$	RCPSP-GPR $m,1,va\|gpr,\rho_i,\delta_i,vr\|C_{\max}$	DTCTP-GPR $1,T\|gpr,\rho_i,\delta_i,disc,mu\|C_{\max}$	DTRTP-GPR $1,1,va\|gpr,\rho_i,\delta_i,disc,mu\|C_{\max}$	MRCPSP-GPR $m,1T,va\|gpr,\rho_i,\delta_i,disc,mu\|C_{\max}$

are classified with respect to whether they allow for generalized precedence relations (CPM precedence constraints, minimal time lags or minimal as well as maximal time lags), multiple activity modes, (multiple) renewable resource types and (multiple) non-renewable resource types (for a definition of renewable and non-renewable resources, we refer the reader to the categorization scheme of Blazewicz et al., 1986). An abbreviation of each resulting problem type is given in the appropriate cell of the table. Table 2 reports the full name of each problem type abbreviation. Using the classification scheme of Herroelen et al. (1998), we have also added the notation of each of the problem types.

The remainder of this paper is organized as follows. In Section 2, we discuss the project representation used. Section 3 is devoted to the description of the MRCPSP-GPR. In Section 4, a brief literature overview is provided. Section 5 elaborates on the temporal analysis of project

Table 2
Abbreviations

Problem type	Description
CPM/PERT	Critical Path Method/Program Evaluation and Review Technique
PDM	Precedence Diagramming Method
MPM	Metra Potential Method
RCPSP	Resource-Constrained Project Scheduling Problem
GRCPSP	Generalized Resource-Constrained Project Scheduling Problem
RCPSP-GPR	Resource-Constrained Project Scheduling Problem with Generalized Precedence Relations
DTCTP	Discrete Time/Cost Trade-off Problem
GDTCTP	Generalized Discrete Time/Cost Trade-off Problem
DTCTP-GPR	Discrete Time/Cost Trade-off Problem with Generalized Precedence Relations
DTRTP	Discrete Time/Resource Trade-off Problem
GDTRTP	Generalized Discrete Time/Resource Trade-off Problem
DTRTP-GPR	Discrete Time/Resource Trade-off Problem with Generalized Precedence Relations
MRCPSP	Multi-Mode Resource-Constrained Project Scheduling Problem
GMRCPSP	Generalized Multi-Mode Resource-Constrained Project Scheduling Problem
MRCPSP-GPR	Multi-Mode Resource-Constrained Project Scheduling Problem with Generalized Precedence Relations

networks in which activity durations are allowed to vary due to multiple execution scenarios. Section 6 discusses the local search methodology that is employed for heuristically solving the MRCPSP-GPR. Computational experience is reported in Section 7. Section 8 is reserved for our conclusions.

2. Project 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 generalized precedence relations (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. Each activity i has M_i execution modes, where each mode determines the activity's duration and its requirements for the set of renewable and non-renewable resources. The duration of an activity in mode m is given by d_{im}. The starting time of an activity i is denoted by s_i and its finishing time by f_i. There are K *renewable* resource types with r_{ikm} denoting the resource requirements of activity i in mode m with respect to resource type k and a_k the availability of resource type k. There also are $\bar{K}$ *non-renewable* resource types with $r_{i\bar{k}m}$ denoting the resource requirements of activity i in mode m with respect to resource type $\bar{k}$ and $a_{\bar{k}}$ the availability of resource type $\bar{k}$.

The activities are subject to generalized precedence relations (GPRs), i.e. arbitrary minimal and maximal time lags between the start and completion of the activities. We distinguish between four types of GPRs: start–start (SS), start–finish (SF), finish–start (FS) and finish–finish (FF). A minimal time lag specifies that an activity can only start (finish) when the predecessor activity 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 certain number of time periods beyond the start (finish) of another activity. GPRs are often useful in practice, for instance in cases where activities require fixed or simultaneous starting or completion times, non-delay execution, mandatory overlaps with other activities, time-varying resource requirements or ready times and deadlines. The first treatment of GPRs is due to Kerbosh 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), Brinkmann and Neumann (1996), Zhan (1994), Neumann and Zhan (1995), Schwindt (1996), De Reyck and Herroelen (1996b, 1998a, b), Schwindt and Neumann (1996), Franck and Neumann (1996), Neumann and Schwindt (1997), De Reyck (1998), De Reyck et al. (1998), Schwindt (1998), Möhring et al. (1998) and Dorndorf and Pesch (1998). The minimal and maximal time lags between two activities i and j have the form:

$$s_i + SS_{ij}^{\min} \leqslant s_j \leqslant s_i + SS_{ij}^{\max},$$
$$s_i + SF_{ij}^{\min} \leqslant f_j \leqslant s_i + SF_{ij}^{\max},$$
$$f_i + FS_{ij}^{\min} \leqslant s_j \leqslant f_i + FS_{ij}^{\max},$$
$$f_i + FF_{ij}^{\min} \leqslant f_j \leqslant f_i + FF_{ij}^{\max},$$

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 $SS_{ij}^{\max}$, $FS_{ij}^{\min}, \ldots$). Each type of GPR can be transformed into a GPR of another arbitrary type. To ensure that the dummy start and finish activities correspond to the beginning and the completion of the project, we assume that the GPRs ensure that activity 1 should always start before every other activity and that activity n can never terminate before any other activity. If this is not the case, we can insert additional zero-lag $SS_{1i}^{\min}$ precedence relations and/or zero-lag $FS_{in}^{\min}$ precedence relations.

3. Problem formulation

Assume that all maximal time lags are transformed into equivalent minimal time lags with a negative value in the opposite direction. For instance, a $FS_{ij}^{\max}$ lag is transformed into a $SF_{ji}^{\min}$ time lag. Define E_{SS} as the resulting set of SS-precedence relations. Similarly, define E_{SF}, E_{FS} and E_{FF}.

The actual values for each of the time lags (whether they originate from a minimal time lag or a maximal time lag) are given by $SS_{ij}, SF_{ij}, FS_{ij}$, and FF_{ij}. The objective is to schedule each activity in one of its modes, subject to GPRs and the resource constraints under the objective of minimizing the project duration. Introducing the decision variables

$$x_{imt} = \begin{cases} 1 & \text{if activity } i \text{ is performed in mode } m \\ & \text{and started at time } t, \\ 0 & \text{otherwise,} \end{cases}$$

the MRCPSP-GPR (denoted as $m,1T,va|gpr, \rho_i,\delta_i,disc,mu|C_{\max}$ in the classification scheme of Herroelen et al., 1998) can be formulated as follows:

$$\min \sum_{t=es_n}^{ls_n} t x_{n1t} \tag{1}$$

subject to

$$\sum_{m=1}^{M_i}\sum_{t=es_i}^{ls_i} x_{imt} = 1, \quad i = 1,2,\ldots,n, \tag{2}$$

$$\sum_{m=1}^{M_i}\sum_{t=es_i}^{ls_i} (t + SS_{ij}) x_{imt} \leqslant \sum_{m=1}^{M_j}\sum_{t=es_j}^{ls_j} t x_{jmt}, \ (i,j) \in E_{ss}, \tag{3}$$

$$\sum_{m=1}^{M_i}\sum_{t=es_i}^{ls_i} (t + SF_{ij}) x_{imt} \leqslant \sum_{m=1}^{M_j}\sum_{t=es_i}^{ls_j} (t + d_{jm}) x_{jmt}, \quad (i,j) \in E_{SF}, \tag{4}$$

$$\sum_{m=1}^{M_i}\sum_{t=es_i}^{ls_i} (t + d_{im} + FS_{ij}) x_{imt} \leqslant \sum_{m=1}^{M_j}\sum_{t=es_j}^{ls_j} t x_{jmt}, \quad (i,j) \in E_{FS}, \tag{5}$$

$$\sum_{m=1}^{M_i}\sum_{t=es_i}^{ls_i} (t + d_{im} + FS_{ij}) x_{imt} \leqslant \sum_{m=1}^{M_j}\sum_{t=es_j}^{ls_j} (t + d_{jm}) x_{jmt}, \quad (i,j) \in E_{FF}, \tag{6}$$

$$\sum_{i=1}^{n}\sum_{m=1}^{M_i} r_{imk} \sum_{s=\max\{t-d_{im},es_i\}}^{\min\{t-1,ls_i\}} x_{ims} \leqslant a_k, \quad k = 1,2,\ldots,K, \ t = 1,2,\ldots,T, \tag{7}$$

$$\sum_{i=1}^{n}\sum_{m=1}^{M_i} r_{im\bar{k}} \sum_{t=es_i}^{ls_i} x_{imt} \leqslant a_{\bar{k}}, \quad \bar{k} = 1,2,\ldots,\bar{K}, \tag{8}$$

$$x_{imt} \in \{0,1\}, \quad i = 1,2,\ldots,n, \quad m = 1,2,\ldots,M_i, \ t = 0,1,\ldots,T, \tag{9}$$

with T an upper bound on the project duration, es_i (ls_i) the earliest (latest) start time of activity i (the calculation of appropriate values for es_i and ls_i will be discussed later). The objective function (1) minimizes the project duration. Constraints (2) ensure that each activity is assigned exactly one mode and exactly one start time. Constraints (3)–(6) denote the GPRs. The resource constraints are given in Eqs. (7) and (8) for renewable and non-renewable resources, respectively. Eq. (9) force the decision variables to assume binary values.

The MRCPSP-GPR, as a generalization of the RCPSP ($m,1|cpm|C_{\max}$), is known to be NP-hard. Also the problem of determining whether an MRCPSP-GPR instance has a feasible solution, is NP-complete, since the MRCPSP-GPR is also a generalization of the RCPSP-GPR ($m,1,va|gpr, \rho_i,\delta_i,vr|C_{\max}$), for which the feasibility problem is proven to be NP-complete (Bartusch et al., 1988).

4. Review of the literature

To the best of our knowledge, the literature on the MRCPSP-GPR is completely void. Solution procedures have been presented either for the MRCPSP with zero-lag finish–start precedence constraints or for the RCPSP-GPR in which every activity is assigned a fixed duration and a fixed set of resource requirements.

Exact procedures for the MRCPSP have been presented by Talbot (1982), Patterson et al. (1989), Speranza and Vercellis (1993), Sprecher (1994), Ahn and Erengüç (1995), Sprecher et al. (1997), Nudtasomboon and Randhawa (1997) and Sprecher and Drexl (1998). Heuristic solution procedures for the MRCPSP have been developed by Talbot (1982), Drexl and Grünewald (1993), Slowinski et al. (1994), Boctor (1993, 1996), Boctor (to appear), Özdamar and Ulusoy (1994), Kolisch (1995), Yang and Patterson (1995), Ahn and

Erengüç (1996), Özdamar (1996), Kolisch and Drexl (1997), Hartmann (1998), Sung and Lim (1997) and Hartmann and Drexl (to appear).

Exact algorithms for the RCPSP-GPR have been presented by Bartusch et al. (1988), De Reyck and Herroelen (1998a, b) and De Reyck et al. (1998b), Schwindt (1998), Möhring et al. (1998) and Dorndorf and Pesch (1998). Heuristic procedures have been given by Zhan (1994), Neumann and Zhan (1995), Brinkmann and Neumann (1996), Schwindt and Neumann (1996), Franck and Neumann (1996) and Schwindt (1998).

5. Temporal analysis

A schedule $S = (s_1, s_2, \ldots, s_n)$ is an array of activity starting times. A mode assignment $\mu = (m_1, m_2, \ldots, m_n)$ assigns a specific mode to each activity and consequently, given a schedule $S = (s_1, s_2, \ldots, s_n)$, yields an array of activity finish times $(f_1, f_2, \ldots, f_n)$. A schedule $S = (s_1, s_2, \ldots, s_n)$ with associated activity finish times $(f_1, f_2, \ldots, f_n)$ is called *time-feasible* if the activity start and finish times satisfy the following conditions:

$$s_i \geqslant 0 \quad \forall i \in V, \tag{10}$$

$$s_i + SS_{ij} \leqslant s_j \quad \forall (i,j) \in E_{SS}, \tag{11}$$

$$s_i + SF_{ij} \leqslant f_j \quad \forall (i,j) \in E_{SF}, \tag{12}$$

$$f_i + FS_{ij} \leqslant s_j \quad \forall (i,j) \in E_{FS}, \tag{13}$$

$$f_i + FF_{ij} \leqslant f_j \quad \forall (i,j) \in E_{FF}, \tag{14}$$

$$\exists m \leqslant M_i : d_{im} = f_i - s_i \quad \forall i \in V, \tag{15}$$

where Eqs. (10) ensure that no activity starts before the current time (time zero), Eqs. (11)–(14) denote the GPRs transformed into minimal time lags and Eqs. (15) represent the activity mode constraints. The minimum starting times $(s_1, s_2, \ldots, s_n)$ satisfying Eqs. (10)–(15)) form the *early start schedule* $ESS = (es_1, es_2, \ldots, es_n)$. When the activity durations are fixed (only one mode per activity), computing an *ESS* is straightforward. The various GPRs can be transformed into minimal start–start precedence relations l_{ij}. In this way, all GPRs are consolidated in the expression $s_i + l_{ij} \leqslant s_j$, where l_{ij} denotes a minimal start–start time lag. Then, the earliest start of an activity i can be calculated by finding the longest path d_{1i} from node 1 to node i in the network defined by the l_{ij} values. The calculation of the matrix $D = [d_{ij}]$ can be done by standard graph algorithms for longest paths in networks, for instance by the Floyd–Warshall algorithm (see Lawler, 1976).

When activity durations are allowed to vary due to multiple activity modes, the temporal analysis is not as straightforward. In that case, the longest path between activities may depend on the selected mode for each of the activities in the project. In the MRCPSP with zero-lag finish–start precedence relations $(m,1T|cpm,disc,mu|C_{\max})$, an *ESS* can be easily derived by assigning each activity its shortest possible mode and then performing a classic CPM analysis. When GPRs (even of the minimal lag type only) are introduced, however, selecting the shortest mode for each activity will not necessarily minimize the project duration. Certain activities in project networks with GPRs may be backward-critical (reverse critical), implying that decreasing their duration leads to an increase of the project duration. Determining the optimal mode assignment that minimizes the project duration may prove to be very difficult.

6. Solution methodology

6.1. Local search methodology

The proposed local search methodology is an extension of the one used in De Reyck et al. (1998a) for solving the discrete time/resource trade-off problem $(1,1|cpm,disc,mu|C_{\max})$, the latter, however, not being able to cope with generalized precedence relations, multiple resource types and non-renewable resources. We will only briefly describe the main features of our approach. More details can be found in De Reyck and Herroelen (1997) and De Reyck (1998). We divide the MRCPSP-GPR into two distinct problems, which are solved in two successive phases: a mode assignment phase and a resource-constrained project

scheduling phase with fixed mode assignments. The *mode assignment phase* assigns to each activity i a specific execution mode m_i. Each mode assignment $\mu = (m_1, m_2, \ldots, m_n)$ then yields a resource-constrained project scheduling problem with generalized precedence relations (RCPSP-GPR), which is subsequently solved in the *resource-constrained project scheduling phase*. A similar local search methodology has been devised by Kolisch and Drexl (1997) for dealing with the MRCPSP $(m,1T|cpm,disc,mu|C_{\max})$. The authors solve the MRCPSP using biased random sampling and a priority-based heuristic for heuristically solving the RCPSP $(m,1|cpm|C_{\max})$ for a fixed mode assignment. For heuristically solving the RCPSP-GPR, we use a truncated version of the branch-and-bound procedure developed by De Reyck and Herroelen (1998a). The algorithm is truncated after a very small amount of time has elapsed (namely when 10 backtracking steps have been performed, which requires, on the average, less than 0.01 s).

We have experimented with several solution approaches, including truncated enumeration, random enumeration, descent methods and a tabu search procedure. A *truncated enumeration procedure systematically* enumerates a number of mode assignments and solves each RCPSP-GPR instance using the truncated RCPSP-GPR procedure. A *random enumeration procedure* differs in that it randomly generates a number of mode assignments which are subsequently evaluated. Local search methods start with an initial mode assignment $\mu = (m_1, m_2, \ldots, m_n)$ and compute an upper bound on the project duration using the RCPSP-GPR procedure. An improvement procedure is then initiated which evaluates (using the same RCPSP-GPR procedure) a number of new mode assignments in the neighbourhood of μ (all mode assignments μ_k in which exactly one activity is assigned another mode) and selects one of them for further exploration. The procedures then perform a *move* from one mode assignment to another one in which exactly one activity is assigned another mode. This process continues until some termination criterion is met. Upon finding the best mode assignment, the same RCPSP-GPR procedure is used, but it is now truncated after 1 second of CPU-time, which results in high quality heuristic solutions as has been shown by De Reyck and Herroelen (1998a).

Given a mode assignment μ, a *steepest descent* method evaluates all mode assignments in the neighbourhood of μ and selects the one with the smallest project duration. A *fastest descent* algorithm differs in that the first encountered improving mode assignment is implemented. Both types of procedures terminate when no improving mode assignment can be found. The descent procedures can be enhanced with *random restarts*.

6.2. A tabu search procedure

6.2.1. Short-term search strategy

Descent methods only accept alterations which result in an improvement of the incumbent solution. As a consequence, a major drawback is their tendency to being trapped in a local optimum. A *tabu search* (*TS*) procedure overcomes this disadvantage by also allowing for non-improving moves and a (temporary) deterioration of the objective function. When no improving moves can be detected, a TS procedure selects the least deteriorating move (steepest descent / mildest ascent). In order to avoid cycling, tabu search employs so-called short-term memory to exclude from consideration a specific number of moves. We classify as tabu those moves that *reverse* one of the recently made moves, i.e. after a move from $m_i = x$ to $m_i = y$, we prevent a move from $m_i = z$ to $m_i = x$ for arbitrary values of z. We use a dynamic tabu list the length of which varies randomly in the interval $[\sqrt{n},\ 3\sqrt{n}]$.

Aspiration criteria are used to determine which tabu restrictions should be overridden in order to identify and again make available those moves that may lead to an overall improvement of the objective function or that can never lead to cycling. The tabu status of such moves is not completely revoked, but transformed into a so-called *pending* tabu status (Glover and Laguna, 1993), which means that the move is eligible for selection if no other non-tabu *improving* move exists. The following aspiration criteria have been implemented in the proposed procedure:

- *Global and regional aspiration by objective*: If a move that is classified as tabu would lead to the best solution obtained so far, the tabu status is overridden and the move is implemented. Additionally, if a tabu move would lead to the best possible solution obtained with a specific mode assigned to a specific activity, we override the tabu status of that move.
- *Aspiration by influence*: We define the influence of a move (the induced degree of change on the structure of the incumbent solution) as the absolute value of the difference between the current duration of an activity and its duration in the new mode assignment. We revoke the tabu status of moves of rather low influence, provided that between the time (iteration) the move has been classified as tabu and the current time (iteration), a move of higher influence has been chosen. We also favour influential moves by making them more attractive in the move selection process.
- *Aspiration by search direction*: If the current (tabu) move results in an improved solution, and if the most recent move out of the new (tabu) mode assignment was also an improving move, the tabu status of the current move is revoked.
- *Aspiration by strong admissibility*: A move is labelled *strongly admissible* if it is eligible to be selected and does not rely on any aspiration criterion to qualify for admissibility, or if it would lead to the best solution obtained so far (Glover, 1990). If a strongly admissible move was made prior to the most recent iteration during which a non-improving move has been made, we revoke the tabu status of every improving move.

The procedure is terminated when (a) 10,000 iterations are performed, or (b) 1000 iterations are performed without improving the best known solution (this number is deemed to be sufficient for the procedure to have either reached the global optimum or to have converged to and being stuck in a local optimum), or (c) the time limit is exceeded, or (d) a solution is encountered with a project duration equal to a lower bound.

6.2.2. Long-term search strategy

The core of a TS procedure is a steepest descent/mildest ascent procedure supplemented with (short-term) recency-based memory in the form of a tabu list to prevent cycling. Although this basic scheme, supplemented with appropriate aspiration criteria, may already outperform pure descent procedures, another component is necessary that typically operates on a somewhat longer time horizon to ensure that the search process examines solutions throughout the entire solution space (*diversification*) and that promising regions of the solution space (good local optima) are examined more thoroughly (*intensification*). This component can be supplied by using *frequency-based memory*. Essentially, frequency-based memory stores information about the frequency that a specific solution characteristic occurs over all generated solutions or about the frequency that a move with a specific attribute has occurred. For instance, we can store (a) the number of generated solutions in which a specific activity was executed in a specific mode, (b) the number of times a move occurred in which an *arbitrary* new mode was reassigned to a specific activity, or (c) the number of times a *specific* mode was assigned to a specific activity.

In the proposed TS procedure, we use frequency-based memory to detect whether the search space has been confined to a small region of the entire solution space, and use that information to guide the procedure into new unexplored regions. We modify the attractiveness of the moves under consideration by including a frequency-based component which makes moves containing frequently encountered attributes less attractive than moves which contain rarely encountered attributes. The diversifying influence on the move selection process is restricted to those occasions when no admissible improving moves exist.

The search process is also divided in different phases, which diversify or intensify the search. After an initial data collection phase in which the required data for computing the frequencies is stored, a diversification phase is started based on frequency information about the number of times that an activity was assigned a specific mode. If the frequencies indicate that for a specific activity only a small subset of all possible modes have been

assigned to that activity, the search space is restricted by excluding those moves that assign one of these modes to that activity. After such a diversification phase, all frequency-based memory is erased and a new data collection is initiated.

Diversification phases are alternated with intensification phases, in which the search is concentrated on promising regions of the solution space. This is done by storing the number of times a specific mode was assigned to each activity. When a high frequency count for a specific activity-mode combination is combined with a small associated project duration, it may be advantageous to "fix" the mode assignment of that activity to one mode or a small subset of all possible modes. The intensification procedure examines all residence frequencies of the previously saved high quality local optima (defined as having an upper bound on the project duration equal to the current best solution) and detects which activities have been assigned a specific mode or a small subset of all possible modes in each or a large number of these solutions. Subsequently, the search space is restricted by limiting the possible modes for each activity to that small subset.

6.3. Reducing the search space using preprocessing

Before initiating the local search procedures, the project data is modified in order to reduce the search space. The following reduction scheme is a modified version of the method devised by Kolisch (1995) for the MRCPSP, with the major difference that when dealing with GPRs, inefficient modes cannot be eliminated from consideration. A mode is called efficient if every other mode has either a higher duration or a higher resource requirement. As mentioned earlier, shortening activities in project networks with GPRs may cause an increase of the project duration. Similarly, prolonging activities may cause a decrease in project duration. Therefore, it is possible that the optimal solution can only be obtained by using inefficient modes.

First, all doubly constrained resource types are replaced by two new resource types, one renewable and the other non-renewable. Then, the feasibility of the problem is examined with respect to both the renewable and the non-renewable resource constraints. If there are activities that require more of a renewable resource than what is available ($\exists i \in V$ and $k \leqslant K$: $\min_{m=1..M_i} r_{imk} > a_k$), the problem is infeasible. Similarly, if the sum of the minimum requirements for a non-renewable resource type exceeds its availability ($\exists \bar{k} \leqslant \bar{K}$: $\sum_{i=1}^{n} \min_{m=1..M_i} r_{im\bar{k}} > a_{\bar{k}}$), the problem is also infeasible.

Subsequently, a number of modes and resource types are eliminated because they play no role whatsoever in the determination of the optimal solution. First, all *non-executable modes* are eliminated. A non-executable mode m_i of activity i inevitably results in a violation of a renewable resource constraint ($\exists k \leqslant K: r_{im_i k} > a_k$) or non-renewable resource constraint ($\exists \bar{k} \leqslant \bar{K}: r_{im_i \bar{K}} > a_{\bar{k}} - \sum_{j=1, j\neq i}^{n} \min_{m=1..M_j} \{r_{jm\bar{k}}\}$). First, all non-executable modes with respect to a renewable resource type are eliminated, because the removal of such a mode may cause other (prior executable) modes to become non-executable with respect to a non-renewable resource type.

Subsequently, all *redundant non-renewable resource types* are eliminated. A non-renewable resource type $\bar{k}$ is called redundant if no mode assignment can result in a violation of the corresponding resource constraint, i.e. if $\sum_{i=1}^{n} \max_{m=1..M_i} \{r_{im\bar{k}}\} \leqslant a_{\bar{k}}$. This rule also eliminates non-renewable resources the demand of which does not depend on the selected activity modes.

6.4. Determining a feasible initial solution

Determining a feasible starting solution for the MRCPSP-GPR is not straightforward, due to the fact that (a) when two or more non-renewable resource types are present, the mode assignment problem is NP-complete (Kolisch and Drexl, 1997) and (b) the feasibility version of the RCPSP-GPR is NP-complete. To determine a feasible initial mode assignment with respect to the non-renewable resource constraints, we use a modified version of the heuristic proposed by Kolisch and Drexl (1997) which assigns modes to activities based on their requirements for each of the non-

renewable resource types. The heuristic goes as follows. Define, for each non-renewable resource type, a residual availability $a_{\bar{k}}^{\text{res}} = a_{\bar{k}} - \sum_{i \in A} r_{im_i\bar{k}}$, where the set A includes all activities i which have already been assigned a mode m_i. Initially, $A = \phi$ and $a_{\bar{k}}^{\text{res}} = a_{\bar{k}}$. Then, compute, for each activity and each of its modes, the relative resource consumption of non-renewable resources as follows: $r_{im}^{\text{avg}} = \sum_{\bar{k}=1}^{\bar{K}} r_{im\bar{k}}/a_{\bar{k}}^{\text{res}}$. For each activity, the mode m_i^* is determined which leads to the least reduction of the available non-renewable resources. Therefore, select for each activity i the mode m_i^* for which $r_{im_i^*}^{\text{avg}} = \sum_{\bar{k}=1}^{\bar{K}} r_{im_i^*\bar{k}}/a_{\bar{k}}^{\text{res}}$ is minimal. Ties are broken according to activity duration (smallest mode first). Then, assign mode m_i^* to activity i for which $r_{im_i^*}^{\text{avg}} = \max_{j \in V \setminus A}\{r_{jm_j^*}^{\text{avg}}\}$. In other words, select the activity for which the mode which puts least mortgage on the non-renewable resources requires more, on the average, of these resources relative to (the corresponding modes of) the other activities. As a tie-breaker, use the activity with the smallest label. As a consequence, the residual availabilities of the non-renewable resources are decreased as soon as possible with an unavoidable amount, thereby making the subsequent decisions more precise.

If there are no non-renewable resources, the heuristic will select, for each activity, the mode with smallest associated activity duration. In the case that only a single non-renewable resource is present, the heuristic always produces a feasible mode assignment, provided one exists. When two or more non-renewable resources are present, no feasible mode assignment can be guaranteed. If in that case, the heuristic fails to produce a feasible mode assignment, we propose to use a truncated implicit enumeration of mode assignments until a feasible mode assignment is encountered. The enumeration is implicit because certain mode assignments can be dominated if, for at least one non-renewable resource type, it can be shown that the resource requirements of the activities in their currently allocated mode together with the theoretical minimum resource consumption of the activities for which a mode has not yet been assigned exceeds the availability.

The explicit checking of this dominance rule can be avoided if preprocessing is applied on the non-renewable resource requirements. For each activity i and non-renewable resource $\bar{k}$, define a value $x_{i\bar{k}}$ as: $x_{i\bar{k}} = \min_{m=1..M_i} r_{im\bar{k}}$. Then, subtract the value $x_{i\bar{k}}$ from the corresponding resource requirements of activity i in every one of its modes: $\forall m = 1..M_i: r_{im\bar{k}} = r_{im\bar{k}} - x_{i\bar{k}}$. This reduces the minimum requirement of each activity for each non-renewable resource type to zero: $\forall i = 1..n, \bar{k} = 1..\bar{K}$: $min_{m=1..M_i} r_{im\bar{k}} = 0$. Finally, the availabilities of each non-renewable resource type have to be adjusted as follows: $\forall \bar{k} = 1..\bar{K}: a_{\bar{k}} = a_{\bar{k}} - \sum_{i \in V} x_{i\bar{k}}$.

If a feasible mode assignment (w.r.t. the non-renewable resource constraints) is encountered, no guarantee can be given that the corresponding RCPSP-GPR has a feasible solution. Since determining whether an RCPSP-GPR instance has a feasible solution constitutes an NP-complete problem, determining a mode assignment that leads to a feasible RCPSP-GPR instance is very hard. However, even if no feasible solution to the RCPSP-GPR can be detected, the local search can be started as long as a feasible RCPSP-GPR solution in the neighbourhood of the initial mode assignment can be found. If not, a random restart can be initiated.

It is clear that determining a feasible initial solution can be very hard. Because our approach is a heuristic one, it is possible that the problem has a feasible solution, but that the procedure cannot detect one. This is the case when (a) the implicit enumeration of mode assignments is truncated before a feasible mode assignment w.r.t. the non-renewable resource constraints is found or (b) the branch-and-bound procedure of De Reyck and Herroelen (1998a) cannot find a feasible solution for the initial RCPSP-GPR instance or for any of the neighbouring RCPSP-GPR instances before it is truncated, and a random restart does not resolve this problem. Several modifications can be used in order to increase the chances of reaching a feasible (initial) solution. First, the implicit enumeration of mode assignments can be prolonged (possibly even to complete enumeration), and second, another procedure can be used to determine a feasible solution for the corresponding RCPSP-GPR instance. De Reyck and Herroelen (1998a) have proposed another search strategy for solving the RCPSP-GPR (referred to as the *TWS*-approach) which involves looking for a feasible solution first

instead of directly trying to locate the optimal solution. This approach greatly improves the chances of finding a feasible solution, whereas the time required for finding and verifying the optimal solution remains more or less the same. The solution quality of the heuristic solutions obtained by truncating the search after a small amount of time is slightly worse compared to the original approach. Using this approach in an MRCPSP-GPR instance for which a feasible solution cannot be determined may very well lead to a feasible solution. Although there is a theoretic possibility that the proposed local search procedures are unable to find a feasible solution for an MRCPSP-GPR instance for which a feasible solution exists, the computational experiments in Section 6 reveal that most procedures succeed in finding a feasible solution for all instances but the ones for which infeasibility has been proven. These results are certainly encouraging.

6.5. Infeasible moves

Moves are classified as *infeasible* when (a) the new mode assignment is infeasible with respect to the non-renewable resource constraints, (b) the resulting RCPSP-GPR instance has no feasible solution or (c) although the resulting RCPSP-GPR has a feasible solution, none can be found by the truncated branch-and-bound procedure within the given time limit. When no feasible moves can be found, the fastest and steepest descent procedures terminate. The iterated descent methods and the TS restart with a randomly determined initial mode assignment. Notice again that a move can be classified as infeasible because the truncated branch-and-bound algorithm cannot determine a feasible schedule based on the new mode assignment, despite the fact that there may well exist a feasible project schedule. This, of course, may lead to missing the optimal solution if (a) the new mode assignment was the optimal one, or (b) the new mode assignment was the only way of reaching the optimal mode assignment. Because there are typically several ways of reaching the optimal mode assignment, we deem case (b) to pose less of a problem. Again, the *TWS*-approach may be used in order to increase the chances of finding a feasible solution for an RCPSP-GPR instance.

6.6. Lower bounds

We use a lower bound on the project makespan which is the maximum of a precedence-based lower bound and a resource-based lower bound. The resource-based lower bound lb_r is computed as $lb_r = \max_{k=1}^{m}\{\lceil(\sum_{i=1}^{n}\min_{m=1}^{M_i} d_{im}r_{imk})/a_k\rceil\}$, where $\lceil x\rceil$ denotes the smallest integer equal to or greater than x. As mentioned in Section 5, the temporal analysis in project networks in which activity durations are allowed to vary is not as straightforward as in the case with fixed activity durations. Determining the optimal mode assignment that minimizes the project duration (even in the absence of resource constraints) constitutes a very hard problem, which requires the use of enumeration techniques. However, this would put a mortgage on the use of the resulting project duration as a lower bound on the optimal project duration for the resource-constrained case. It is essential that a lower bound can be computed very efficiently. Therefore, for the sake of computing a lower bound, we assume that the activity durations can be selected from a continuous interval rather than their discrete set of possible values. Consequently, it is possible to efficiently compute a lower bound on the project duration. First, we compute for each activity i the minimal duration $d_i^{\min} = \min_{m=1}^{M_i}\{d_{im}\}$ and maximal duration $d_i^{\max} = \max_{m=1}^{M_i}\{d_{im}\}$ that can be assigned to that activity. Then, we assume that the duration of each activity i can vary continuously between the interval $[d_i^{\min}, d_i^{\max}]$. The following algorithm then computes optimal activity durations and a minimal project duration, which can serve as a lower bound on the duration of the project with discrete activity durations and for the resource-constrained case.

Step 1. Initialization

Transform all maximal time lags into negative minimal time lags in the opposite direction.

Define E_{SS} as the resulting set of SS-precedence relations. Similarly, define E_{SF}, E_{FS} and E_{FF}. $\forall i \in V \setminus \{1\}$, set es_i and ef_i to -9999. Set $es_1 = 0$ and $ef_1 = 0$.

Set IT, the number of iterations to 0.

Step 2. Determine earliest starting and finishing times

Increase IT by 1.

For each activity $i \in V$ do

{Determine $S_{SS} = \{j \in S | (j,i) \in E_{SS}\}, S_{SF} = \{j \in S | (j,i) \in E_{SF}\}$, $S_{FS} = \{j \in S | (j,i) \in E_{FS}\}$ and $S_{FF} = \{j \in S | (j,i) \in E_{FF}\}$.

Compute $es_i = \max\{\{es_j SS_{ij} | j \in S_{SS}\}, \{ef_i + FS_{ij} | j \in S_{FS}\}\}$ and $ef_i = \max\{\{es_j + SF_{ij} | j \in S_{SF}\}, \{ef_j + FF_{ij} | j \in S_{FF}\}\}$.

If $ef_i - es_i > b_i$, set $es_i = ef_i - b_i$, else if $ef_i - es_i < a_i$, set $ef_i = es_i + a_i$.

}

Step 3. Repeat until no change occurs in the labels or a positive cycle is detected

If $IT > n$, STOP and report that the project is time-infeasible.

If for any activity i, es_i or ef_i has changed during the previous iteration, go to STEP 2.

Otherwise, report the minimal project makespan ef_n and STOP.

The minimal project duration ef_n can be used as a *modified precedence-based lower bound* lb_p^{mod}. The lower bound for the MRCPSP-GPR then equals $lb = \max\{lb_r, lb_p^{\text{mod}}\}$. The algorithm above can also detect time-infeasibility. In that case, the es_i and/or the ef_i values in Step 2 will be modified over and over again, making the procedure cycle forever until it is halted. We know that the maximum number of iterations (IT, the number of times Step 2 is executed) equals n. Therefore, if IT exceeds n, infeasibility is detected and the procedure is terminated. In that case, the original project network with discrete durations is also infeasible.

6.7. Mode-dependent generalized precedence relations

So far we assumed that the GPRs are independent of the modes of the associated activities. Actually, in some real-life project scheduling applications, it may occur that the value of a time lag depends on the duration of either the predecessor or successor activity. For an example we refer to De Reyck (1998). The proposed procedures are able to cope with such mode-dependent GPRs under the assumption that (a) the time lags only vary with the duration of the originating (predecessor) activity and (b) although the time lags are mode-dependent, the nature of the precedence relation is not. Assumption (a) can easily be relaxed if precedence relations from an activity i to another activity j, the time lag of which depends on the duration (mode) of activity j, are transformed into precedence relations from j to i. Assumption (b) seems very reasonable. In our experience, it seems highly illogical that the nature of a precedence relation depends on the duration of the associated activities.

The major modification in the procedures in order to cope with mode-dependent GPRs concerns the computation of the lower bound. In the precedence-based lower bound lb_p^{mod}, the values for the time lags used in the calculations should be based on their minimal value, i.e. based on the mode for the originating activity for which the time lag is minimal. This will result in slightly inferior lower bound values.

7. Computational experience

The procedures have been programmed in Microsoft Visual C^{++} 2.0 under Windows NT for use on a Digital Venturis Pentium-60 personal computer. The codes themselves require at most 105kb and the data structures use at most 1.4Mb of internal memory, which allows them to be used on computer platforms with little available memory. A time limit of 100 s is imposed.

7.1. Benchmark problem set

We used ProGen/max (Schwindt, 1996) to generate 1350 MRCPSP-GPR instances using a full factorial experimental design by varying several parameters as given in Table 3. 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 the experiment. Table 4 shows the parameter settings of the full factorial experiment.

Table 3
The parameter settings of the benchmark problem set

Control parameter	Value
n	10; 20; 30
M (number of modes)	1; 2; 3; 4; 5
d_i	[1, 10]
Number of initial and terminal activities	[2, 3]
Maximal number of predecessors and successors	3
OS (order strength)	0.50
Degree of redundancy	0.0
Percentage of maximal time lags	[10%, 20%]
Number of cycle structures	[1, 10]
Number of activities per cycle structure	[1, 30]
Coefficient of cycle structure density	0.3
Deviations of minimal time lags from duration	0.5
Tightness of maximal time lags	0.5
Slack factor	0.0
K (number of renewable resource types)	2
$\bar{K}$ (number of non-renewable resource types)	2
r_{imk} (renewable resource demand)	[1, 10]
r_{imk} (non-renewable resource demand)	[1, 10]
RF_{ren} (resource factor for renewable resources)	1
RF_{non} (resource factor for non-renewable resources)	1
RS_{ren} (resource strength for renewable resources)	0.25; 0.50; 0.75
RS_{non} (resource strength for non-renewable resources)	0.25; 0.50; 0.75

For each combination of the control parameter values, 10 problem instances have been generated.

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 $n(n-1)/2$ (Mastor, 1970). Because OS only applies to acyclic networks, it is computed based on the acyclic skeleton of the project networks obtained by removing all the maximal time lags. The *degree of redundancy* (Schwindt, 1996) is computed by dividing the number of redundant arcs in the network by their theoretical maximal value. A *cycle structure* (Zhan, 1994) is defined as a strongly connected component of a project network with GPRs. It contains a number of activities and directed arcs (minimal time lags originating from standardizing the GPRs) such that there is a directed path from each activity to each other activity. The *coefficient of cycle structure density* (Schwindt, 1996) is a measure of the amount of precedence relations in a cycle structure. The higher the number of precedence relations, the more dense the cycle structure. The *deviations of the minimal time lags from the activity durations* (Schwindt, 1996) determine how the values of the minimal time lags relate to the activity durations. The *tightness of maximal time lags* (Schwindt, 1996) determines how the values of the maximal time lags relate to their theoretical minimal value which preserves time-feasibility and to a maximum value which always results in time- and resource-feasibility. The *slack factor* (Schwindt, 1996) determines how the minimal time lags depend on the activity modes. The higher the slack factor, the higher the dependency of the minimal time lags on the associated activity modes.

The *resource factor RF* (Pascoe, 1966) reflects the average portion of resources requested per activity. A resource factor is defined for both non-renewable and renewable resource types:

Table 4
The parameter settings of the full factorial experiment

Control parameter	Value
n (number of activities)	10; 20; 30
M (number of modes)	1; 2; 3; 4; 5;
RS_{ren}	0.25; 0.50; 0.75
RS_{non}	0.25; 0.50; 0.75

$$RF_{\text{ren}} = \frac{1}{nK}\sum_{i=1}^{n}\frac{1}{M_i}\sum_{m=1}^{M_i}\sum_{k=1}^{K}\begin{cases}1 & \text{if } r_{imk} > 0,\\ 0 & \text{otherwise,}\end{cases}$$

$$RF_{\text{non}} = \frac{1}{n\bar{K}}\sum_{i=1}^{n}\frac{1}{M_i}\sum_{m=1}^{M_i}\sum_{\bar{k}=1}^{\bar{K}}\begin{cases}1 & \text{if } r_{im\bar{k}} > 0,\\ 0 & \text{otherwise.}\end{cases}$$

The *resource strength for the renewable resource types*, RS_{ren}, is defined by Kolisch et al. (1995) as $\left(a_k - r_k^{\min}\right)/\left(r_m^{\max} - r_m^{\min}\right)$, where $r_k^{\min} = \max_{i=1..n}\{\min_{m=1..M_i} r_{imk}\}$, and $r_k^{\max}$ is the peak demand for renewable resource type k in the precedence-based *ESS* based on the activity modes with maximal resource requirement and on the minimal time lags only. Hence, with respect to one resource the smallest feasible resource availability is obtained for $RS_{\text{ren}} = 0$. For $RS_{\text{non}} = 1$, the problem is no longer resource-constrained. Similarly, the *resource strength for the non-renewable resource types*, RS_{non}, is defined as $\left(a_{\bar{k}} - r_{\bar{k}}^{\min}\right)/r_{\bar{k}}^{\max} - r_{\bar{k}}^{\min}$, where $r_{\bar{k}}^{\min} = \sum_{i=1}^{n}\min_{m=1..M_i} r_{im\bar{k}}$, and $r_{\bar{k}}^{\max} = \sum_{i=1}^{n}\max_{m=1..M_i} r_{im\bar{k}}$.

In an attempt to reduce the size of the experiment, we fixed certain problem parameters at specific values rather than varying them over the entire range of complexity. We fixed *OS* at 0.5 and RF_{ren} and RF_{non} at 1. Setting *RF* at 1 leads to a bias towards the more difficult problem instances, since the complexity of resource-constrained project scheduling problems is known to increase when the *RF* increases (Kolisch et al., 1995; De Reyck and Herroelen, 1998a). In total, 36 instances of the 1350 problems turn out to be infeasible.

7.2. Basic computational results

Tables 5 and 63 summarize our findings. They report the average and maximal deviation with respect to the best known solution, obtained by running all procedures for 1000 s each (the results in Tables 5 and 6 are obtained using a 100 s time limit). Also given are the number of times the best known solution is obtained, the number of problems solved to optimality, the average number of RCPSP-GPR instances solved and the average required CPU-time (in seconds). Table 5 reports the average results on the entire problem set, whereas Table 6 focuses on the 90 largest instances with 30 activities and 5 modes per activity.

With the exception of truncated enumeration, all procedures succeed in finding a feasible solution for all problem instances except for those that are known to be infeasible. Truncated enumeration does not find a feasible solution for 110 problem instances, including 74 instances for which a feasible solution is known.

The proposed TS procedure clearly is more effective than its competing local search methods. For about 92% of the instances, TS is able to match the best known solution (obtained by running all procedures, *including the TS*, for 1000 s). However, the maximum deviation with respect to

Table 5
Summary results

	Truncated enumeration	Random enumeration	Fastest descent	Steepest descent	Iterated fastest descent	Iterated steepest descent	Tabu search	Hybrid tabu search
Avg. dev. from best sol.	41.77%	19.40%	11.90%	13.44%	1.27%	2.52%	0.45%	0.40%
Max. dev. from best sol.	353.85%	133.33%	247.37%	247.37%	91.67%	141.67%	35.90%	20.00%
Best solution	554 (41%)	594 (44%)	656 (49%)	631 (47%)	1,141 (85%)	1,041 (77%)	1,246 (92%)	1,232 (91%)
Optimal	388 (29%)	371 (27%)	442 (33%)	406 (30%)	593 (44%)	556 (41%)	603 (45%)	603 (45%)
RCPSP-GPRs solved	55,811	28,380	102	382	11,788	15,994	5,883	5,732
Avg. CPU-time (s)	65.51	72.00	1.46	5.49	46.02	54.46	44.35	43.40

Table 6
Results for the instances with 30 activities and 5 modes

	Truncated enumeration	Random enumeration	Fastest descent	Steepest descent	Iterated fastest descent	Iterated steepest descent	Tabu search	Hybrid tabu search
Avg. dev. from best sol.	167.59%	79.85%	26.89%	42.66%	8.43%	18.42%	3.44%	2.55%
Max. dev. from best sol.	316.67%	133.33%	247.37%	247.37%	91.67%	141.67%	35.90%	17.39%
Best solution	0 (0%)	0 (0%)	13 (14%)	4 (4%)	36 (40%)	10 (11%)	54 (60%)	66 (73%)
Optimal	0 (0%)	0 (0%)	11 (12%)	4 (4%)	24 (27%)	9 (10%)	27 (30%)	27 (30%)
RCPSP-GPRs solved	9498	4999	354	1525	2920	4967	3875	3637
Avg. CPU-time (s)	101.00	99.24	6.41	30.07	58.08	86.00	85.81	80.81

the best known solution equals 35.90%. This value is caused by an instance with 30 activities and 5 modes per activity. Clearly, this indicates that further improvements are possible. Truncated and random enumeration perform very poorly, although random enumeration seems to outperform truncated enumeration. Truncated enumeration suffers from the fact that while enumerating all possible mode assignments in a systematic way, resource feasibility with respect to the non-renewable resources may not be attained. Also, a random enumeration results in a much more diversified search. Fastest and steepest descent also perform rather bad. Enhancing the descent procedures with random restarts substantially improves the quality of the obtained solutions, at the expense of increased computation times. Iterated fastest descent turns out to be quite effective. Contrary to expectations, the overall quality of the solutions obtained with steepest descent are inferior to those obtained using a fastest descent approach.

Inspired by the relatively good results obtained with the iterated fastest descent approach, we developed a hybrid TS algorithm which takes advantage of both steepest and fastest descent. The original TS is modified in the sense that when a mode assignment is encountered that results in the best solution obtained so far, it is immediately enforced. In other words, we use a fastest descent each time the best solution obtained so far is improved upon. A steepest descent is used when the procedure finds improving moves which do not lead to the best known solution. As was the case in the original algorithm, a mildest ascent is used when no improving moves can be found. The major differences in behaviour of the original and the hybrid TS algorithm are:

- The original TS uses steepest descent, whereas the hybrid TS uses fastest descent until the first local optimum is encountered.
- The original TS investigates all mode assignments in the neighbourhood of the current mode assignment. Even if a move is found that improves the best known solution, the search continues for even better moves. The hybrid approach immediately accepts the first move leading to a global improvement, thereby possibly missing better paths of improvement.
- The original TS considers the tabu status of moves, even if they improve the best known solution. Although in that case, the *global aspiration by objective* aspiration criterion is applicable, the corresponding moves are classified as *pending* tabu moves which are only considered when no other feasible improving non-tabu moves can be found. Contrary, the hybrid approach accepts a global improvement regardless of its tabu status.

The results indicate that, on the average, the hybrid TS performs better than the original TS, especially for the larger instances (Table 6).

7.3. An extended time limit

Although TS outperforms all other local search methods, it does not succeed in finding the best

solution for some of the instances. Detailed results reveal that for these instances, 100 seconds is not enough to exploit the full advantages of the intensification and diversification processes. Tables 7 and 8 report the results when the CPU-time limit is increased to 1000 s. Clearly, an increase of the time limit allows the TS to substantially increase its effectiveness. For the largest instances (Table 8), the average (maximal) deviation from the best known solution is decreased from 3.44% (35.90%) to 0.03% (2.70%). The hybrid TS procedure now performs slightly worse than the original approach. The relative performance of TS versus the other local search methods is improved significantly. The other local search methods do not benefit greatly from an increased allowed CPU-time. Only iterated steepest descent takes advantage of the extra available time and performs almost equally as good as a fastest descent approach. To summarize, an increased time limit results in an improved performance for procedures based on steepest descent (iterated steepest descent and the original TS). Iterated fastest descent and the hybrid TS do not gain as much. Clearly, steepest descent is very time-consuming compared to fastest descent (see Table 5), and should only be undertaken if enough time is available.

7.4. Detailed computational results

Detailed computational results showing the impact of problem size and other problem characteristics on the performance of the local search methods are given in De Reyck and Herroelen (1997). The main results can be summarized as follows:

Table 7
Results for the complete problem set – 1000 s time limit

	Truncated enumeration	Random enumeration	Fastest descent	Steepest descent	Iterated fastest descent	Iterated steepest descent	Tabu search	Hybrid tabu search
Avg. dev. from best sol.	35.99%	15.35%	11.90%	13.44%	0.92%	1.10%	0.09%	0.13%
Max. dev. from best sol.	280.00%	134.38%	247.37%	247.37%	133.33%	96.55%	15.00%	15.00%
Best solution	652(48%)	643(48%)	656 (49%)	631 (47%)	1,228 (91%)	1,183 (88%)	1,318 (98%)	1,293 (96%)
Optimal	445 (33%)	398 (29%)	442 (33%)	406 (30%)	607 (45%)	597 (44%)	612 (45%)	610 (45%)
RCPSP-GPRs solved	533,458	302,513	102	382	34,270	54,845	14,787	18,592
Avg. CPU-time (s)	569.24	696.82	1.46	5.49	278.38	352.21	201.08	205.50

Table 8
Results for the instances with 30 activities and 5 modes – 1000 s time limit

	Truncated enumeration	Random enumeration	Fastest descent	Steepest descent	Iterated fastest descent	Iterated steepest descent	Tabu search	Hybrid tabu search
Avg. dev. from best sol.	152.68%	65.22%	26.89%	42.66%	7.61%	9.89%	0.03%	0.52%
Max. dev. from best sol.	266.67%	134.38%	247.37%	247.37%	133.33%	96.55%	2.70%	12.50%
Best solution	0 (0%)	0 (0%)	13 (14%)	4 (4%)	52 (58%)	35 (39%)	89 (99%)	73 (81%)
Optimal	0 (0%)	0 (0%)	11 (12%)	4 (4%)	26 (29%)	22 (24%)	29 (32%)	27 (30%)
RCPSP-GPRs solved	114,825	57,701	354	1,525	24,815	35,599	29,563	28,281
Avg. CPU-time (s)	1000.03	943.03	6.41	30.07	431.14	574.43	616.49	602.31

- TS outperforms all the other heuristic procedures for each problem class, independently of the size and characteristics of the problem instances.
- RS_{ren} has a negative impact on the computational complexity of the MRCPSP-GPR. The higher RS_{ren}, the easier the corresponding problem instance. These results are in line with results reported in the literature for related problem types (Kolisch et al., 1995; De Reyck and Herroelen, 1998a; Schwindt, 1998). However, we conjecture that if a more refined experiment is undertaken (with values for RS_{ren} between 0 and 0.2), a bell-shaped effect of RS_{ren} on the computational complexity of the MRCPSP-GPR will be revealed. This bell-shaped effect has already been conjectured by Elmaghraby and Herroelen (1980) for the RCPSP and verified by De Reyck and Herroelen (1998a) for the RCPSP and by De Reyck et al. (1998b) for the RCPSP-GPR.
- The effect of RS_{non} on the complexity of the MRCPSP-GPR is U-shaped. When RS_{non} is of medium value (0.50), the problems seem to be relatively easy. When RS_{non} is higher (0.75) and especially when RS_{non} is low (0.25), the problems becomes much harder. The reason for this U-shaped effect stems from two different sources. On the one hand, a decreasing RS_{non} makes the mode assignment problem more complex in the sense that it is harder to find feasible mode assignments. On the other hand, when RS_{non} is rather high, only very few mode assignments can be eliminated because of the non-renewable resource requirements. Therefore, much more mode assignments need to be examined. Consequently, the neighbourhood used in the local search methods expands substantially. This results in a longer time required for scanning the neighbourhood and a less thorough examination of the solution space.

8. Conclusions

We presented a local search methodology and a tabu search procedure for solving a general type of resource-constrained project scheduling problem, embodied in the multi-mode resource-constrained project scheduling problem with generalized precedence relations (MRCPSP-GPR). The MRCPSP consists of a number of project activities, subject to generalized precedence relations (minimal and maximal time lags between the activity starting and completion times), which require a specific amount of possibly several renewable, non-renewable and doubly constrained resources for which a limited availability is imposed. The activities possess different execution modes, which reflect different ways of performing the activity, each mode possibly having a different impact on the duration of the activity, the costs associated with the activity and the required use of resources. These multiple modes give rise to several kinds of trade-offs (time/resource, time/cost and resource/resource trade-offs) which allow for a more efficient use of the resources.

The proposed local search methodology divides the MRCPSP-GPR into two distinct phases: a mode assignment phase and a resource-constrained project scheduling phase with fixed mode assignments. Several solution methods have been examined, among which are truncated and random enumeration, descent methods and a full-fledged tabu search procedure. The results indicate that the tabu search procedure is capable of outperforming all other heuristic approaches. A study on the impact of several problem characteristics on the computational complexity of the MRCPSP-GPR reveals that a decreasing availability of the renewable resources leads to a more complex MRCPSP-GPR instance. The effect of the availability of the non-renewable resources is U-shaped: a medium value corresponds to MRCPSP-GPR instances which are the hardest to solve.

It should be clear that the MRCPSP-GPR is a very difficult problem to solve. Not only is the problem NP-hard, but also the two related subproblems, namely the mode assignment problem and the resource-constrained project scheduling problem with generalized precedence relations, are NP-hard. This complexity is apparent from the absence of exact procedures for this problem type. Moreover, the NP-hardness of the two subproblems creates difficulties in determining feasible

solutions. Sometimes, the procedures proposed in this paper may be unable to provide a feasible solution without verifying that indeed no feasible solution exists at all. Although the computational results were encouraging in this respect, examining other approaches for dealing with infeasibility undoubtedly constitutes a promising area of future research.

Acknowledgements

We would like to thank the anonymous referees for their helpful suggestions. We are also indebted to Klaus Neumann and Christoph Schwindt from the University of Karlsruhe for providing us with the project generator ProGen/max.

References

Ahn, T., Erengüç, S.S., 1995. Resource-constrained project scheduling with multiple crashable modes: An exact solution method, INFORMS New Orleans Fall Meeting, 29 October–1 November.

Ahn, T., Erengüç, S.S., 1996. The resource-constrained project scheduling problem with multiple crashable modes: A heuristic procedure. Proceedings of the Fifth International Workshop on Project Management and Scheduling Poznan, April 11–13, 23–26.

Bartusch, M., Möhring, R.H., Radermacher, F.J., 1988. Scheduling project networks with resource constraints and time windows. Annals of Operations Research 16, 201–240.

Blazewicz, J., Cellary, W., Slowinski, R., Weglarz, J., 1986. Scheduling under resource constraints – deterministic models. Annals of Operations Research 7.

Boctor, F., 1993. Heuristics for scheduling projects with resource restrictions and several resource-duration modes. International Journal of Production Research 31, 2547–2558.

Boctor, F., 1996. A new and efficient heuristic for scheduling projects with resource restrictions and multiple execution modes. In: Herroelen, W., Demeulemeester, E. (Eds.), European Journal of Operational Research, Special Issue on Project Management and Scheduling, vol. 90, pp. 349–361.

Boctor, F., to appear. An adaptation of the simulated annealing algorithm for solving resource-constrained project scheduling problems. International Journal of Production Research.

Brinkmann, K., Neumann, K., 1996. Heuristic procedures for resource-constrained project scheduling with minimal and maximal time lags: the minimum project-duration and resource-levelling problem. Journal of Decision Systems 5, 129–156.

Crandall, K.C., 1973. Project planning with precedence lead/lag factors. Project Management Quarterly 4, 18–27.

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., Herroelen, W., 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., Herroelen, W., 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., Herroelen, W., 1998a. A branch-and-bound algorithm for the resource-constrained project scheduling problem with generalized precedence relations. European Journal of Operational Research 111, 152–174.

De Reyck, B., Herroelen, W., 1998b. An optimal procedure for the resource-constrained project scheduling problem with discounted cash flows and generalized precedence relations. Computers and Operations Research 25, 1–17.

De Reyck, B., Demeulemeester, E., Herroelen, W., 1998a. Local search methods for the discrete time/resource trade-off problem in project networks. Naval Research Logistics 45, 553–578.

De Reyck, B., Demeulemeester, E., Herroelen, W., 1998b. Algorithms for scheduling projects with generalized precedence relations. Chapter 4 in: Weglarz, J. (Ed.), Project Scheduling – Recent Models, Algorithms and Applications, International Series in Operations Research and Management Science, vol. 14. Kluwer Academic Publishers, Dordrecht, pp. 77–106.

Dorndorf, U., Pesch, E., 1998. Constraint-based project scheduling. Proceedings of the 16th European Conference on Operational Research, 12–15 July, Brussels.

Drexl, A., Grünewald, J., 1993. Non-preemptive multi-mode resource-constrained project scheduling. IIE Transactions 25, 74–81.

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

Elmaghraby, S.E., Herroelen, W., 1980. On the measurement of complexity in activity networks. European Journal of Operational Research 5, 223–234.

Elmaghraby, S.E., Kamburowski, J., 1992. The analysis of activity networks under generalized precedence relations. Management Science 38, 1245–1263.

Franck, B., Neumann, K., 1996. Priority-rule methods for the resource-constrained project scheduling problem with minimal and maximal time lags – an empirical analysis. Proceedings of the Fifth International Workshop on Project Management and Scheduling, April 11–13, Poznan, 88–91.

Glover, F., Laguna, M., 1993. Tabu search. In: Reeves, C. (Ed.), Modern Heuristic Techniques for Combinatorial Problems, Blackwell, Oxford.

Glover, F., 1990. Tabu search: A tutorial. Interfaces 20, 74–94.

Hartmann, S., 1998. A competitive genetic algorithm for resource-constrained project scheduling, Naval Research Logistics 45, 733–750.

Hartmann, S., Drexl, A., to appear. Project scheduling with multiple modes: A comparison of exact algorithms. Networks.

Herroelen, W., Demeulemeester, E., De Reyck, B., 1998. A classification scheme for project scheduling. Chapter 1 in: Weglarz, J. (Ed.), Project Scheduling – Recent Models, Algorithms and Applications, International Series in Operations Research and Management Science, vol. 14. Kluwer Academic Publishers, Dordrecht, pp. 1–26.

Kerbosh, J.A.G.M., Schell, H.J., 1975. Network planning by the Extended METRA Potential Method. Report KS-1.1, University of Technology Eindhoven, Department of Industrial Engineering.

Kolisch, R., 1995. Project scheduling under resource constraints – Efficient heuristics for several problem cases, Physica-Verlag, Heidelberg.

Kolisch, R., Drexl, A., 1997. Local search for non-preemptive multi-mode resource-constrained project scheduling. IIE Transactions 29, 987–999.

Kolisch, R., Sprecher, A., Drexl, A., 1995. Characterization and generation of a general class of resource-constrained project scheduling problems. Management Science 41, 1693–1703.

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. Management Science 16, 728–746.

Moder, J.J., Phillips, C.R., Davis, E.W., 1983. Project Management with CPM, PERT and Precedence Diagramming, 3rd edn. Van Nostrand Reinhold Company, New York.

Möhring, R.H., Stork, F., Uetz, M., 1998, Resource-constrained project scheduling with time windows: A branching scheme based on dynamic release dates. Proceedings of the Sixth International Workshop on Project Management and Scheduling, Istanbul, July 7–9, 98–101.

Neumann, K., Schwindt, C., 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., Zhan, J., 1995. Heuristics for the minimum project-duration problem with minimal and maximal time lags under fixed resource constraints. Journal of Intelligent Manufacturing 6, 145–154.

Nudtasomboon, N., Randhawa, S.U., 1997. Resource-constrained project scheduling with renewable and non-renewable resources and time/resource trade-offs. Computers and Industrial Engineering 32, 227–242.

Özdamar, L., Ulusoy, G., 1994. A local constraint based analysis approach to project scheduling under general resource constraints. European Journal of Operational Research 79, 287–298.

Özdamar, L., 1996. A genetic algorithm approach for the multi-mode project scheduling problem under general resource categories. Proceedings of the Fifth International Workshop on Project Management and Scheduling. Poznan, April 11–13, 178–185.

Pascoe, T.L., 1966. Allocation of resources – CPM. Revue Française de Recherche Opérationelle 38, 31–38.

Patterson, J.H., Slowinski, R., Talbot, F.B., Weglarz, J., 1989. An algorithm for a general class of precedence and resource constrained scheduling problems. Chapter 1 in: Slowinski, R., Weglarz, J. (Eds.). Advances In Project Scheduling. Elsevier, Amsterdam.

Roy, B., 1962. Graphes et ordonnancement. Revue Française de Recherche Opérationelle, 323–333.

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., Neumann, K., 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, April 11–13, Poznan, 212–215.

Schwindt, C., 1998. Verfahren zur Lösung des ressourcenbeschränkten Projectdauerminimierungsproblems mit planungsabhängingen Zeitfenstern, Shaker-Verlag, Aachen.

Slowinski, R., Soniewicki, B., Weglarz, J., 1994. DSS for multi-objective project scheduling. European Journal of Operational Research 79, 220–229.

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

Sprecher, A., 1994. Resource-Constrained Project Scheduling – Exact Methods for the Multi-Mode Case, Lecture Notes in Economics and Mathematical Systems. Springer, Berlin.

Sprecher, A., Drexl, A., 1998. Solving multi-mode resource-constrained project scheduling problems by a simple, general and powerful sequencing algorithm. European Journal of Operational Research 107, 431–450.

Sprecher, A., Hartman, S., Drexl, A., 1997. An exact algorithm for project scheduling with multiple modes. OR Spektrum 19, 195–203.

Sung, C.S., Lim, S.K., 1997. A scheduling procedure for a general class of resource-constrained projects. Computers and Industrial Engineering 32, 9–17.

Talbot, F.B., 1982. Resource-constrained project scheduling problem with time-resource trade-offs: The nonpreemptive case. Management Science 28, 1197–1210.

Wiest, J.D., 1981. Precedence diagramming methods: some unusual characteristics and their implications for project managers. Journal of Operations Management 1, 121–130.

Yang, K.K., Patterson, J.H., 1995. Scheduling a resource-constrained project with alternative performance modes using simulated annealing. INFORMS New Orleans Fall Meeting, 29 October–1 November.

Zhan, J., 1994. Heuristics for scheduling resource-constrained projects in MPM networks. European Journal of Operational Research 76, 192–205.