ELSEVIER

European Journal of Operational Research 112 (1999) 3–41

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

Invited Review

# Resource-constrained project scheduling: Notation, classification, models, and methods

Peter Brucker [a,1], Andreas Drexl [b,*], Rolf Möhring [c,2], Klaus Neumann [d,3], Erwin Pesch [e,4]

[a] *Universität Osnabrück, Fachbereich Mathematik/Informatik, Albrechtstr. 28, 49069 Osnabrück, Germany*
[b] *Universität Kiel, Institut für Betriebswirtschaftslehre, Olshausenstr. 40, 24118 Kiel, Germany*
[c] *Technische Universität Berlin, Fachbereich Mathematik, Straße des 17. Juni, 10623 Berlin, Germany*
[d] *Universität Karlsruhe, Institut für Wirtschaftstheorie und Operations Research, Kaiserstr. 12, 76128 Karlsruhe, Germany*
[e] *Universität Bonn, Institut für Gesellschafts- und Wirtschaftswissenschaften, Adenauerallee 24-42, 53113 Bonn, Germany*

Received 1 June 1998

**Abstract**

Project scheduling is concerned with single-item or small batch production where scarce resources have to be allocated to dependent activities over time. Applications can be found in diverse industries such as construction engineering, software development, etc. Also, project scheduling is increasingly important for make-to-order companies where the capacities have been cut down in order to meet lean management concepts. Likewise, project scheduling is very attractive for researchers, because the models in this area are rich and, hence, difficult to solve. For instance, the resource-constrained project scheduling problem contains the job shop scheduling problem as a special case. So far, no classification scheme exists which is compatible with what is commonly accepted in machine scheduling. Also, a variety of symbols are used by project scheduling researchers in order to denote one and the same subject. Hence, there is a gap between machine scheduling on the one hand and project scheduling on the other with respect to both, viz. a common notation and a classification scheme. As a matter of fact, in project scheduling, an ever growing number of papers is going to be published and it becomes more and more difficult for the scientific community to keep track of what is really new and relevant. One purpose of our paper is to close this gap. That is, we provide a classification scheme, i.e. a description of the resource environment, the activity characteristics, and the objective function, respectively, which is compatible with machine scheduling and which allows to classify the most important models dealt with so far. Also, we propose a unifying notation. The second purpose of this paper is to review some of the recent developments. More specifically, we review exact and heuristic algorithms for the single-mode and the multi-mode case, for the time–cost tradeoff problem, for problems with minimum and maximum time lags, for problems with other objectives than makespan minimization and, last but not least, for problems with stochastic activity durations. 

* Corresponding author. Fax: +49-431-880-1531; e-mail: drexl@bwl.uni.kiel.de
[1] E-mail: peter@mathematik.uni-osnabrueck.de
[2] E-mail: moehring@math.tu-berlin.de
[3] E-mail: neumann@wior.uni-karlsruhe.de
[4] E-mail: E.Pesch@uni-bonn.de

0377-2217/99/$ – see front matter 
*PII* S0377-2217(98)00204-5

*Keywords:* Project scheduling/resource constraints; Notation; Classification scheme; Single-mode case; Time–cost tradeoffs; Multi-mode case; Minimum and maximum time lags; Nonregular objectives; Stochastic activity durations; Constraint propagation

## 1. Scope and purpose

Project scheduling has attracted an ever growing attention in recent years both from science and practice. It is concerned with single-item or small batch production where scarce resources have to be met when scheduling dependent activities over time. Project scheduling is important for make-to-order companies where the capacities have been cut down in order to cope with lean management concepts. Project scheduling is very attractive for researchers also, because the models in this area are rich in the sense that many well-known optimization problems are special cases of the more general project scheduling models. For instance, the resource-constrained project scheduling problem contains the job shop scheduling problem as a special case. Without surprise, project scheduling problems in general are really challenging from a computational point of view.

Both practice and science of project scheduling have evolved fast recently, producing numerous acronyms to distinguish between different problem classes. Also, a variety of symbols are used by project scheduling researchers in order to denote one and the same subject. Hence, sometimes it is difficult to keep a clear view of what the subject is all about, because the models in this area are not standardized. Recently Herroelen et al. [94] made a first attempt to provide a classification scheme for project scheduling. Unfortunately, their scheme is not compatible with what is commonly accepted in machine scheduling. Hence, there is still a gap between machine scheduling on the one hand and project scheduling on the other with respect to both, viz. a common notation and a classification scheme. One purpose of our paper is to close this gap. We provide a classification scheme, i.e. a description of the resource environment, the activity characteristics, and the objective function, respectively, which is compatible with machine scheduling and which allows to classify the most important models dealt with so far. Also, we propose a unifying notation.

Another purpose of this paper is to review some of the recent developments. [5] Additional surveys have been given by, e.g., [67,92,113,151].

The paper is organized as follows. In Section 2 the notation and the classification scheme are introduced. Section 3 covers exact and heuristic algorithms for the single-mode resource-constrained project scheduling problem. In Section 4 we review solution procedures for the time–cost tradeoff problem. The multi-mode resource-constrained project scheduling problem is the subject of Section 5. In Section 6 we concentrate on resource-constrained project scheduling in the presence of minimum and maximum time lags. Section 7 is dedicated to problems with nonregular objective functions. Section 8 discusses models with stochastic activity durations. We conclude the paper in Section 9 with an exposition of further models.

Our work will illustrate that there are numerous different models each of which requires tailored methods in order to cope with their inherent (computational) complexity. Nevertheless, constraint propagation techniques have recently evolved, the aim of which is to solve a variety of constraint satisfaction problems. Although their impact on the field of resource-constrained (project) scheduling still is not clear we decided to add an Appendix A the subject of which are constraint propagation-based sequence consistency tests.

## 2. Notation and classification scheme

In the last few years many different problems in project scheduling have been considered and time

[5] Most of the working papers covered in this review are available on the internet via http://www.wior.uni-karlsruhe.de/rcpsp/.

is ready to have a unified notation and a general classification scheme for project scheduling. It is important that this scheme is compatible with what is generally accepted in machine scheduling (cf. [77]) and resource-constrained machine scheduling (cf. [18]), because machine scheduling models are special cases of project scheduling models.

In the sequel we will first propose a unifying notation. Basically, we assume a project to consist of activities (jobs) $1,\ldots,n$. For the sake of simplicity, in general a unique dummy beginning activity 0 and a unique dummy termination activity $n+1$ are added. Frequently, the structure of the project is depicted by a so-called activity-on-node (AON) network where the nodes and the arcs represent the activities and the precedence relations, respectively. $G=(V,E)$ denotes the graph of precedence constraints (transitively reduced), while single precedence constraints are denoted alternatively by $i \rightarrow j$ or $(i,j)$. $Pred(j)$ defines the set of direct predecessors while $Succ(j)$ is the set of direct successors of activity $j$. The processing time of activity $j$ is given by $p_j$.

There is a set $\mathscr{R}^{\rho}$ of renewable, a set $\mathscr{R}^{v}$ of nonrenewable and, possibly, a set of doubly constrained resources. Renewable means that a prespecified number of units of a resource is available for every period of the planning horizon $T$. Nonrenewable says that a number of units of a resource is available for the entire planning horizon. As usual, we skip the notion of doubly constrained resources, because they can be covered by the renewable and the nonrenewable ones. The per period usage of activity $j$ of renewable resource $k$ is denoted by $r^{\rho}_{jk}$ while $R^{\rho}_{k}$ defines the (constant) number of units of resource $k$ available in every period. In the multi-mode case, $\mathscr{M}_j$ defines the set of modes, that is, processing alternatives of activity $j$. The processing time of activity $j$ in mode $m$ is given by $p_{jm}$. The per period usage (total resource consumption) of activity $j$ of renewable (nonrenewable) resource $k$ is given by $r^{\rho}_{jkm}$ ($r^{v}_{jkm}$) while $R^{v}_{k}$ defines the number of units of nonrenewable resource $k$ available for the entire planning horizon. In the single-mode case, that is for $|\mathscr{M}_j|=1$ for all $j$ and $\mathscr{R}^{v}=\emptyset$, we skip the mode index $m$ and the superscript $\rho$ for the sake of simplicity.

$S_j$ ($C_j$) denotes the start time (completion time) of activity $j$. Consequently, $S=(S_1,\ldots,S_n)$ is a schedule and $C=(C_1,\ldots,C_n)$ is the vector of completion times. $\mathscr{S}_{\mathrm{T}}$ defines the set of time-feasible schedules, $\mathscr{S}_{\mathrm{R}}$ the set of resource-feasible schedules and $\mathscr{S}=\mathscr{S}_{\mathrm{R}}\cap\mathscr{S}_{\mathrm{T}}$ the set of feasible schedules. $t$ is an index for time periods. Finally, $d^{\min}_{ij}$ and $d^{\max}_{ij}$ denote minimum and maximum time lags, respectively, between the start of activities $i$ and $j$. In general, parameters are assumed to be integer-valued.

Table 1 summarizes the notation introduced along with some minor additions.

Now we extend the $\alpha|\beta|\gamma$-scheme used in the machine scheduling literature.

*$\alpha$: Resource environment:* To distinguish between specific machine scheduling problems and project scheduling problems we introduce in the $\alpha$-field *PS* (project scheduling) or *MPS* (multi-mode project scheduling). *PS* can be augmented to *PS* $m,\sigma,\rho$ according to the notation of Blażewicz et al. [18] for resource-constrained machine scheduling. In the case of multi-mode project scheduling also nonrenewable resources may be considered. In this case the notation is analogously augmented by $MPSm,\sigma,\rho;\mu,\tau,\omega$.

| | |
|---|---|
| *PS* | project scheduling |
| *MPS* | multi-mode project scheduling |
| $PSm,\sigma,\rho$ | $m$ resources, $\sigma$ units of each resource available, each activity requires at most $\rho$ units of the resources |
| $MPSm,\sigma,\rho;\mu,\tau,\omega$ | multi-mode project scheduling with $m$ renewable resources, $\sigma$ units of each resource available, each activity requires at most $\rho$ units of the resources, $\mu$ nonrenewable resources, $\tau$ units of each resource available, each activity requires at most $\omega$ units of the resources |

If an entry of $m,\sigma,\rho;\mu,\tau,\omega$ is replaced by $\cdot$, the values of the parameters are specified in the input. For $PSm,\cdot,\cdot$ and $PSm,\sigma,\cdot$ we write *PSm* and $PSm,\sigma$, respectively, for short. If all values in $m,\sigma,\rho$ are specified in the input, we write $\cdot$ instead of $\cdot,\cdot,\cdot$.

Table 1
Basic notation

| Symbol | Definition |
|---|---|
| $V$ | set of activities |
| $n$ | number of real activities |
| $E$ | set of precedence or temporal constraints |
| $G = (V, E)$ | directed graph of precedence or temporal constraints |
| $i \rightarrow j$, $(i, j)$ | precedence constraint |
| $Pred(j)$ | set of direct predecessors of activity $j$ |
| $Succ(j)$ | set of direct successors of activity $j$ |
| $p_j$ | processing time of activity $j$ |
| $\mathscr{R}^{\rho}$ | set of renewable resources |
| $R_k^{\rho}$ | constant amount of available units of renewable resource $k$ |
| $r_{jk}^{\rho}$ | per period usage of activity $j$ of renewable resource $k$ |
| $\mathscr{M}_j$ | set of modes (processing alternatives) of activity $j$ |
| $p_{jm}$ | processing time of activity $j$ in mode $m$ |
| $\mathscr{R}^{v}$ | set of nonrenewable resources |
| $R_k^{v}$ | total amount of available units of nonrenewable resource $k$ |
| $r_{jkm}^{\rho}$ | per period usage of activity $j$ of renewable resource $k$ when processed in mode $m$ |
| $r_{jkm}^{v}$ | consumption of activity $j$ of nonrenewable resource $k$ when processed in mode $m$ |
| $S_j$ | start time of activity $j$ |
| $S = (S_1, \ldots, S_n)$ | schedule |
| $C_j$ | completion time of activity $j$ |
| $C = (C_1, \ldots, C_n)$ | vector of completion times |
| $\mathscr{S}_{\mathrm{T}}$ | set of time-feasible schedules |
| $\mathscr{S}_{\mathrm{R}}$ | set of resource-feasible schedules |
| $\mathscr{S} = \mathscr{S}_R \cap \mathscr{S}_T$ | set of feasible schedules |
| $r_k(S, t)$ | resource consumption of resource $k$ of schedule $S$ at time $t$ |
| $\bar{d}$ | deadline for project duration |
| $T, t$ | time horizon, index for periods |
| $t = 1, 2, \ldots, T$ | periods |
| $[t-1, t[$ | time interval corresponding to period $t$ |
| $d_{ij}^{\min}/d_{ij}^{\max}$ | minimum/maximum time lag between start of activities $i$ and $j$ |

Likewise, for $PS\cdot$ and $MPS\cdot$; $\cdot$ we write $PS$ and $MPS$, respectively.

*Examples:*

| | |
|---|---|
| $PSm, 1, 1$ | $m$ resources, 1 unit of each resource available, each activity requires at most 1 unit of the resources |
| $PSm, \infty$ | $m$ resources, unlimited number of resource units available (i.e., there are no explicit resource constraints, e.g. in resource leveling) |
| $PS1$ | one resource |

*β: Activity characteristics:* We use established notations from machine scheduling (cf. [77]) like $p_j$ processing times, *prec* general precedence constraints, etc.

| | |
|---|---|
| $p_j = 1$ | all processing times (activity durations) are equal to one |
| $p_j = sto$ | stochastic processing times |
| $\bar{d}$ | deadline for project duration |
| *prec* | precedence constraints between activities |
| *chains*, *intree*, *outtree*, *tree* ... | precedence relations between activities are specified by chains, intree, outtree, tree ... |
| *temp* | general temporal constraints given by minimum and maximum start–start time lags between activities |

*γ: Objective function:* As in most cases for machine scheduling we describe objective functions by the corresponding formulas. Besides classical objective functions like $C_{\max}, L_{\max}, \sum w_j C_j$ etc., further criteria may be considered, for example:

| | |
|---|---|
| $\sum c_j^{\mathrm{F}} \beta^{C_j}$ | net present value ($c^{\mathrm{F}}$ cash flow, $\beta$ discount factor) |
| $\sum c_k f(r_k(S,t))$ | resource leveling ($c_k$ cost per unit of resource $k$, $r_k(S,t)$ usage of resource $k$ at time $t$ given schedule $S$) |
| $\sum c_k \max r_k(S,t)$ | resource investment |

Different types of functions $f$ which have been considered in literature and practice will be discussed in Section 7.1.

Some of the models covered in this paper can now be classified as follows:

- $PS \mid prec \mid C_{\max}$: This model forms the core problem among the class of resource-constrained project scheduling problems. Basically, while minimizing the project's makespan, we have to observe precedence and resource constraints. Recently, a couple of papers have contributed new solution procedures. However, the problem is still rather challenging from a computational point of view. Methods for solving this model are reviewed in Section 3.
- $MPS \mid prec \mid C_{\max}$: Models of this class capture resource–resource and time–resource tradeoffs. Hence, they come more close to what can be observed in reality of project management. Methods for solving this model are reviewed in Section 5.
- $PS \mid temp \mid C_{\max}$: In many applications, beside minimum time lags, maximum start–start time lags between activities must be observed. Here, already the feasibility problem is NP-complete in the strong sense. Methods for solving this model are reviewed in Section 6.
- $PS \mid temp \mid \sum c_k f(r_k(S,t))$: In some applications the availability of renewable resources is limited and, in addition, we have to come up with a schedule which levels the resource usage over time. Methods for solving this model and related ones are reviewed in Section 7.

## 3. Single-mode case

In this section enumerative and heuristic methods for solving the basic resource-constrained project scheduling problem $PS \mid prec \mid C_{\max}$ will be summarized. Assume that the project consists of a set $V = \{0, 1, \ldots, n, n+1\}$ of activities where activity $j = 0$ $(j = n+1)$ is a fictitious beginning (termination) activity. The network is assumed to be acyclic and depicted by an activity-on-node network with nodes as activities and arcs as precedence relations. Preemption is not allowed. There are scarce renewable resources. All data are assumed to be integer-valued. The objective is to find a makespan-minimal schedule that meets the constraints imposed by the precedence relations and by limited resource availabilities.

Given an upper bound $T$ on the minimum project duration we can use the precedence relations to derive time windows, i.e. intervals $[EC_j, LC_j]$, with earliest completion time $EC_j$ and latest completion time $LC_j$, containing the precedence feasible completion times of activity $j \in V$, by forward and backward recursion. Analogously, the interval $[ES_j, LS_j]$ bounded from below and above by the earliest start time $ES_j$ and latest start time $LS_j$, respectively, can be calculated to reflect the precedence feasible start times. In general, $PS \mid prec \mid C_{\max}$ is formulated as a 0-1 integer program which makes use of variables $x_{jt} = 1$, if activity $j$ is completed in period $t$ (0, otherwise). Alternatively, it is stated similar to what is presented in Section 6.1 for the more general $PS \mid temp \mid C_{\max}$. For the sake of shortness, we do not present a formal model here.

Section 3.1 describes recent branch-and-bound approaches for $PS \mid prec \mid C_{\max}$ while Section 3.3 surveys heuristics. Lower bounds are important for both types of methods. They are the subject of Section 3.2. In Section 3.4 computational results are briefly discussed.

Within the last years branch-and-cut methods improved the solvability of several combinatorial optimization problems substantially. The generation of valid inequalities and their propagation through an LP solver might be considered as an early effective start of propagation of constraints based on consistency tests. In scheduling successful

applications have been achieved for disjunctive scheduling problems (e.g. job shop scheduling) and are currently going to be extended to $PS \mid prec \mid C_{\max}$. The aim of such efforts is to accelerate heuristics or branch-and-bound methods through an early detection of nonattractive or infeasible nodes. A comprehensive introduction into constraint propagation has been provided by e.g. Tsang [195]. Constraint propagation-based consistency tests are described in Appendix A.

### 3.1. Branch-and-bound methods

Starting with an early work of Johnson [101] a variety of branch-and-bound algorithms have been developed for $PS \mid prec \mid C_{\max}$. Most of them use partial schedules which are associated with the vertices of the enumeration tree. The branching process consists of extending the partial schedule in different ways. Dominance rules, lower bounds, and immediate selection allow to decrease the number of alternatives for extending the partial schedule. The methods use different branching schemes and pruning methods. In general, depth-first-search is used in order to keep memory requirements low.

*The Precedence Tree:* Patterson et al. [155] proposed an algorithm guided by the so-called precedence tree. The procedure begins with starting the dummy beginning activity at time 0. At each level $g$ of the branch-and-bound tree, the set $SJ_g$ of the currently scheduled activities and the set $EJ_g$ of the eligible activities, that is, those activities the predecessors of which are already scheduled, is determined. Then an eligible activity $j_g$ is selected. Now the earliest precedence and resource feasible start time $S_{j_g}$ that is not less than the start time assigned on the previous level of the search tree is computed. Then we branch to the next level. If the dummy termination activity is eligible, a complete schedule has been found. In this case, backtracking to the previous level occurs. Here, the next untested eligible activity is chosen. If all eligible activities have been tested, we track another step back. Each branch from the root to a leaf of the precedence tree corresponds to a permutation of the set of activities which is precedence feasible in the sense that each predecessor of an activity $j_g$ has a smaller index in the sequence than $j_g$. Recently, this algorithm has been enhanced with powerful search tree reduction techniques by Sprecher [185].

*Delay Alternatives:* This algorithm bases on the concept of delay alternatives used by Christofides et al. [40] which has been enhanced by Demeulemeester and Herroelen [48]. In contrast to the precedence tree algorithm, here each level $g$ of the branch-and-bound tree is associated with a fixed time instant $t_g$ (decision point) at which activities may be started. Consequently, a different definition of eligible activities is used in this algorithm: A currently unscheduled activity $j$ is called eligible at time $t_g$ if all of its predecessors $i$ are scheduled with a completion time $C_i \leqslant t_g$. Furthermore, an activity $j$ with start time $S_j$ is said to be in process at time $t_g$ if we have $S_j \leqslant t_g < S_j + p_j$. The proceeding at the current level $g$ of the branch-and-bound tree is as follows: The new decision point $t_g$ is determined as the earliest completion time of the activities in process at $t_{g-1}$. Note that, due to the constant availability levels of the renewable resources, only finish times of scheduled activities need to be considered for starting unscheduled ones. Using the set $FJ_g$ of the activities that are finished at or before the decision point, the set $EJ_g$ of the eligible activities is computed. Having started all eligible activities by adding them to the set $JIP_g$ of the activities in process, may have caused a resource conflict. Thus, the set of the minimal delay alternatives is computed according to the following definition: A delay alternative $\mathscr{DA}_g$ is a subset of $JIP_g$ such that for each renewable resource $k \in \mathscr{R}$ it is $\sum_{j \in JIP_g \setminus \mathscr{DA}_g} r_{jk} \leqslant R_k$. A delay alternative $\mathscr{DA}_g$ is called minimal if no proper subset of $\mathscr{DA}_g$ is a delay alternative. A minimal delay alternative is selected and the activities to be delayed are removed from the current partial schedule. Note, if no resource conflict occurs, the only minimal delay alternative is the empty set. We store the start times of an activity $j$ to be delayed because this information has to be restored during backtracking. Then it is branched to the next level and the next decision point is computed. If the schedule is complete now, backtracking is performed and the next minimal delay alternative is tested. Clearly,

this procedure is different from the precedence tree algorithm in that sets of activities instead of (single) activities are started at each level of the branch-and-bound tree. Moreover, here the time instant at which activities may be started is determined before the activities themselves are selected. Finally, in contrast to the precedence tree algorithm, this approach allows to withdraw scheduling decisions at the current level that have been made at a lower level.

*Extension Alternatives:* Stinson et al. [188] proposed to use extension alternatives to construct partial schedules. As in the previous algorithm, each level $g$ of the branch-and-bound tree is associated with a decision point $t_g$, a set $JIP_g$ of the activities in process, a set $FJ_g$ of the finished activities, and a set $EJ_g$ of eligible activities. Then the current partial schedule is extended by starting a subset of the eligible activities at the decision point without violating the resource constraints. More precisely, an extension alternative $\mathscr{E}\mathscr{A}_g$ is a subset of the eligible set for which $\sum_{j\in JIP_g\cup\mathscr{E}\mathscr{A}_g} r_{jk} \leqslant R_k$ holds for each resource $k \in \mathscr{R}$ and, moreover, $\mathscr{E}\mathscr{A}_g \neq \emptyset$ if $JIP_g = \emptyset$. Note, in order to secure that the algorithm terminates, we may only have non-empty extension alternatives if no activities are in process. However, if there are currently activities in process, the empty set is always an extension alternative which must be tested in order to guarantee optimality. At the current level $g$ of the branch-and-bound tree the procedure is as follows: Determine the new decision point and compute the set of the eligible activities and the set of extension alternatives. Finally, select an extension alternative $\mathscr{E}\mathscr{A}_g$ and start the corresponding activities before branching to the next level. The backtracking mechanism equals the one of the previous algorithm. Note that this procedure is different from the previous algorithm: Whereas the former includes the possibility to delay activities that have been started on a lower than the current level, the latter does not allow to withdraw a scheduling decision of a lower level. As a consequence, we may not restrict the search to "maximal" extension alternatives while we do not lose optimality when considering only minimal delay alternatives. Note, Stinson et al. [188] introduced the procedure solely by means of an example.

*Block Extensions:* Mingozzi et al. [126] consider a slightly different approach based on the following ideas. There exists an optimal schedule defining times

$$t_0 = 0 < t_1 < t_2 < \cdots < t_l$$

and corresponding sets of activities $A_1, \ldots, A_l$ such that

(i) each $t_i (i > 0)$ is the finishing time of some activity,
(ii) all activities in $A_i$ can be processed jointly during $[t_{i-1}, t_i[$ $(i = 1, \ldots, l)$,
(iii) if an activity $j \in A_i$ is not finished in $[t_{i-1}, t_i[$ it will also be processed in $[t_i, t_{i+1}[$, and
(iv) all predecessors of any activity which start at time $t_i$ are scheduled before time $t_i$.

A block consists of such an interval $[t_{i-1}, t_i[$ with a set $A_i$ of activities which can be processed jointly. Furthermore a partial schedule is defined by a sequence of blocks satisfying conditions (iii) and (iv). Then it is branched by adding new blocks providing again partial schedules.

*Schedule schemes:* The branch-and-bound algorithm developed by Brucker et al. [32] generalizes branch-and-bound methods for the job shop scheduling problem and the multiprocessor task scheduling problem (cf. [30,118]). It also uses concepts which can be found in Bartusch et al. [12]. Instead of using partial schedules, sets of feasible schedules are represented by the so-called schedule schemes. Schedule schemes can be motivated as follows.

For two arbitrary activities a schedule $S$ induces either a parallelity relation $i\|j$ or one of the two conjunctions $i \rightarrow j$ or $j \rightarrow i$. $i \rightarrow j$ holds if and only if $i$ finishes before the start time of $j$. $i\|j$ means that $i$ and $j$ are processed in parallel for at least one time unit. We get sets of schedules by relaxing these relations. $i \rightarrow j$ or $j \rightarrow i$ are relaxed by the disjunction $i - j$. $i - j$ means that we have either $i \rightarrow j$ or $j \rightarrow i$. Furthermore disjunctions $i - j$ and parallelity relations $i\|j$ can be relaxed to flexibility relations $i \sim j$. $i \sim j$ means that it is undecided yet which of the two relations $i - j$ or $i\|j$ holds. $\mathscr{C}, \mathscr{D}, \mathscr{N}$ and $\mathscr{U}$ denote the sets of conjunctions, disjunctions, parallelity relations, and flexibility relations, respectively. $(\mathscr{C}, \mathscr{D}, \mathscr{N}, \mathscr{U})$ is a schedule

scheme if for any two different activities $i, j$ exactly one of the following relations holds: $i \to j \in \mathscr{C}$ or $j \to i \in \mathscr{C}$ or $i - j \in \mathscr{D}$ or $i \| j \in \mathscr{N}$ or $i \sim j \in \mathscr{U}$. A schedule scheme $(\mathscr{C}, \mathscr{D}, \mathscr{N}, \mathscr{U})$ defines a set of feasible schedules (which may be empty), namely all feasible schedules which satisfy all the relations in $\mathscr{C}, \mathscr{D}, \mathscr{N}$.

If $\mathscr{C}_0$ is the set of all given precedence relations, $\mathscr{D}_0$ is the set of all pairs of activities which cannot be processed in parallel due to the resource constraints, and $\mathscr{U}_0$ the set of all remaining pairs $i \sim j$, then $(\mathscr{C}_0, \mathscr{D}_0, \emptyset, \mathscr{U}_0)$ represents the set of all feasible schedules. $(\mathscr{C}_0, \mathscr{D}_0, \emptyset, \mathscr{U}_0)$ corresponds to the root of the enumeration tree.

For a schedule scheme of the form $(\mathscr{C}, \mathscr{D}, \mathscr{N}, \emptyset)$ it can be shown that either no feasible schedule satisfying all the relations exists or a dominating feasible schedule can be calculated (cf. [117]). Both can be done in $O(n^3)$ time. Thus, schedule schemes of the form $(\mathscr{C}, \mathscr{D}, \mathscr{N}, \emptyset)$ can be treated as leaves of the enumeration tree and one can branch by replacing a flexibility relation $i \sim j$ by $i - j$ or $i \| j$.

The inclusion of conjunctions in schedule schemes allows to form a temporal analysis in each node of the enumeration tree. By this analysis new conjunctions, disjunctions or parallelity relations are deduced. Furthermore, for the activities time windows can be calculated which improve lower bound calculations.

*Minimal forbidden sets:* Igelmund and Radermacher [99,100] have introduced a branching scheme based on minimal forbidden sets. A more detailed discussion of these concepts can be found in Sections 6.2 and 8.2 of this survey.

Beside lower bounds which are described in Section 3.2, dominance rules are successfully used within e.g. the partial enumeration algorithms of Demeulemeester and Herroelen [48,49] and Sprecher [185] in order to prune large parts of the search tree. Among the most powerful dominance rules is the cutset rule which makes use of stored information about already evaluated partial schedules. During the search process the rule compares the current partial schedule with the stored data. If it can be proven that any solution obtainable from the current partial schedule cannot be better than a solution obtainable from a previously evaluated partial schedule the information of which has been stored, then backtracking may be performed. A description of such rules is skipped here for the sake of shortness and the reader is referred to Section 5.2 where some rules are outlined.

### 3.2. Lower bounds

Usually lower bounds for the optimal solution value of $PS \mid prec \mid C_{\max}$ can be calculated by relaxing some of the constraints and solving the relaxed problem to optimality. Relaxation of the resource constraints leads to the critical path length which provides a simple lower bound. Stinson et al. [188] improve this bound by adding an activity $i$ which does not belong to a critical path $CP$. They calculate a maximal number $e_i$ of time units activity $i$ can be processed in parallel with $CP$. By adding $\max\{0, p_i - e_i\}$ to the critical path length a new lower bound is provided. Instead of adding a single activity, Demeulemeester and Herroelen augment a critical path $CP$ by a path $P$ node-disjoint with $CP$ and calculate a lower bound for $CP \cup P$ using a dynamic programming procedure. In general this lower bound does not coincide with the optimal solution value for $CP \cup P$ (cf. [172]). However, the two-path relaxation can be solved to optimality by a graphical method developed for the job shop problem with two jobs. This approach also allows to find an optimal solution for $CP \cup P$ which respects time windows for the activities in $CP \cup P$ (cf. [28]).

A bound of Mingozzi et al. [126] is based on the following linear program which relaxes partially the precedence constraints and allows preemption. They consider maximal sets of activities which can be processed in parallel. Let $a_1, a_2, \ldots, a_q$ be the characteristic vectors of all these sets. Then the linear programming relaxation has the form

$$\min \quad \sum_{j=1}^{q} x_j \tag{1}$$

$$\text{s.t.} \quad \sum_{j=1}^{q} a_{ij} x_j \geqslant p_i, \quad i = 1, \ldots, n, \tag{2}$$

$$x_j \geqslant 0, \quad j = 1, \ldots, q, \tag{3}$$

where $x_j$ denotes the number of time units all activities represented by $a_j$ are processed jointly. The integer version of the dual of Eqs. (1)–(3) is a set packing problem. Mingozzi et al. [126] provide lower bounds for the resource-constrained project scheduling problem by solving the set packing problem heuristically.

Baar et al. [5] solve the linear program (1)–(3) directly by applying column generation techniques. Although the number of columns is growing exponentially with the number of activities the method is quite fast. Brucker and Knust [31] enhanced the approach by taking into account time windows for the activities. These time windows are derived from the precedence constraints using a fictitious upper bound $T$ for the makespan. Now the columns correspond to sets of activities which can be processed jointly in a given time window. The objective is to find a preemptive schedule respecting all times windows. If such a schedule does not exist, $T$ is a lower bound. Binary search provides largest $T$ with this property. These "destructive improvement" technique has also been used by Klein and Scholl [105] who tested various methods for proving infeasibility of an upper bound $T$.

Other linear programming based bounds have been introduced in connection with the branch-and-bound algorithm of Christofides et al. [40].

### 3.3. Heuristic methods

The first heuristic methods were priority-rule based scheduling methods (cf. [103]). Up to now, a multitude of priority rules were proposed and tested experimentally (cf. [4,22,42,153,194]). Priority-based heuristics have the advantage of being intuitive, easy to implement, and fast in terms of computational effort. However, they do not excel with respect to the average deviation from the optimal objective function value. Hence, recent research interests shifted to more elaborate heuristics like truncated branch-and-bound (cf. [4]), integer programming based heuristics (cf. [148]), disjunctive arc concepts (cf. [4,14]), local constraint-based analysis (cf. [196]), sampling techniques (cf. [110]), and local search techniques (cf. [5,24,85,108,119,169]).

### 3.4. Computational results

Demeulemeester and Herroelen [48] tested their branch-and-bound algorithm on the Patterson-set (cf. [154]) which consists of 110 test problems with up to 51 activities. They solved these problems with an average computation time of 0.21 s (IBM PS/2 Model 70 A21, 25 MHz) outperforming the algorithm of Stinson et al. [188] by a factor of nearly 12.

Kolisch et al. [116] developed the parameter-driven project generator ProGen which thereafter has been widely used as a tool for the evaluation of algorithms proposed for resource-constrained project scheduling. Meanwhile test sets with 30, 60, 90, and 120 activities have been generated each consisting of 480 instances of various types. The 30 activity test set was also used to test the approach by Demeulemeester and Herroelen [48] on a personal computer (IBM PS/2 Model 55SX, 386SX, 15 MHz). Whereas the Patterson-set has been solved within 1.06 s on average, only 415 of the 480 problems instances have been solved within a time limit of 1000 s per problem.

Mingozzi et al. [126] report that their linear programming formulation based bounds perform better than the critical sequence bound introduced by Stinson et al. [188], and that their algorithm is competitive to the procedure presented by Demeulemeester and Herroelen [48], the best one known up to then. Demeulemeester and Herroelen [49] enhanced their approach by adapting a lower bound of Mingozzi et al. [126] and by representing four resources of 8 bit size through one 32 bit unsigned integer. Allowing as much as 24 Mbyte they could solve the entire 30 activity benchmark-set for the first time. The CPU-time averages about 34 s on a personal computer (80486, 25 MHz). Sprecher [185] compared his branch-and-bound algorithm with the enhanced version of the algorithm of Demeulemeester and Herroelen pointing out the tradeoff between computation times and memory requirements. While the enhanced algorithm of Demeulemeester and Herroelen [49] solves 479 of the 480 benchmark problems with 30

activities within average time of 12.33 s using 24 Mbyte memory, the algorithm of Sprecher solves the same problems with average time of 12.85 s using 400 Kbyte memory.

The branch-and-bound algorithm of Brucker et al. [32] solves less problems than other algorithms, but its main advantage is smaller memory requirement. For problems with up to 90 activities it uses at most 10 Mbyte. 326 of the 480 benchmark problems with 60 activities were verified for the first time within a 1 h time limit on a SUN/Sparc 20/801 workstation.

Comparing the heuristic methods it can be concluded that the adaptive search algorithm of Kolisch and Drexl [110], the tabu search method based on schedule schemes of Baar et al. [5], the genetic algorithm of Hartmann [85] and the simulated annealing algorithm of Bouleimen and Lecocq [24] provide the best results. Recently, Kolisch and Hartmann [112] performed a comparison of most of the available heuristic algorithms for larger instances. According to these results, the genetic algorithm of Hartmann [85] and the simulated annealing algorithm of Bouleimen and Lecocq [24] are the most promising candidates.

## 4. Time–cost tradeoff problems

So far, we have only considered project networks with *fixed* (and known) processing times. A generalization of this setting that is still deterministic (and assumes complete information) is obtained by permitting processing times to *vary* according to how much the planner is willing to pay for it. In view of the next section, this control on the processing times can be interpreted as allocation of a nonrenewable resource to the activities, where a larger allocation to an activity (i.e., a higher cost input) reduces its processing time.

The planner then aims at either minimizing the project makespan subject to a fixed upper bound on the nonrenewable resource (the *budget problem*), or at minimizing the total allocation subject to a given bound on the makespan (the *deadline problem*). As the allocation is usually measured in money, these problems are commonly referred to as time–cost tradeoff problems.

### 4.1. Model

A formal model for time–cost tradeoff problems consists of a set $V = \{0, 1, \ldots, n, n+1\}$ of activities, a directed graph $G = (V, E)$ of precedence constraints among the activities, and, for each activity $j$, a set $\mathcal{M}_j$ of possible processing times $p_j \in \mathcal{M}_j$, together with a nonincreasing function $c_j : \mathcal{M}_j \to \mathbb{R}_+$ that models the individual tradeoff between processing time $p_j$ of activity $j$ and the cost (or amount of resource) $c_j(p_j)$ allocated to it. Section 5 covers the model $MPS \mid prec \mid C_{\max}$, a generalization of the model dealt with in this section. There, $\mathcal{M}_j$ will be used in a similar way to denote the set of modes.

Different assumptions on the sets $\mathcal{M}_j$ and the cost functions $c_j$ lead to different subcases of time–cost tradeoff problems.

For instance, if every $\mathcal{M}_j$ is a closed interval $\mathcal{M}_j = [a_j, b_j]$ and $c_j$ is affine linear and decreasing on $\mathcal{M}_j$, we have the *linear time–cost tradeoff problem* introduced by Kelley and Walker [104].

If, on the other hand, every $\mathcal{M}_j$ is a discrete (i.e., finite) set and $c_j$ is decreasing on $\mathcal{M}_j$, we have the *discrete time–cost tradeoff problem* introduced by Harvey and Patterson [87] and Hindelang and Muth [96].

A *realization* $p \in \mathbb{R}_+^{n+2}$ of the project is an assignment of processing times $p_j \in \mathcal{M}_j$ to activities $j \in V$. The *total cost* $c(p)$ of the realization $p$ is given by $c(p) = \sum_{j \in V} c_j(p_j)$. The makespan $C_{\max}(p)$ of the realization $p$ is the makespan of the earliest start schedule of the project when activity $j$ has processing time $p_j$, i.e., the length of a longest path in $G = (V, E)$ with $p_j$ as "length" of vertex $j \in V$.

Fixing either cost or time, we obtain two related optimization problems with the objective to minimize the other parameter.

*Budget problem:* For a given nonnegative budget $b \geqslant 0$, find a realization $p$ with $c(p) \leqslant b$ that minimizes the makespan $C_{\max}(p)$.

*Deadline problem:* For a given deadline $\overline{d}$ on the makespan, find a realization $p$ with $C_{\max}(p) \leqslant \overline{d}$ that minimizes the total cost $c(p)$.

Usually, one wants to solve these problems for all possible budgets or deadlines. This leads to the function

$$T_{\text{opt}}(b) := \min\{C_{\max}(p) \mid p_j \in \mathcal{M}_j,\ c(p) \leqslant b\}$$

giving the minimum makespan as a function of the budget $b$, and the function

$$B_{\text{opt}}(\overline{d}) := \min\{c(p) \mid p_j \in \mathcal{M}_j,\ C_{\max}(p) \leqslant \overline{d}\},$$

giving the minimum cost as a function of the deadline $\overline{d}$ (the *project cost curve*).

The budget problem is a special case of $MPS1 \mid prec \mid C_{\max}$ which is covered in Section 5. While in this section we have no renewable resources and only one single nonrenewable resource, Section 5 considers the general multi-mode version. Similarly, the deadline problem combines characteristics of the multi-mode case (Section 5) and of the resource levelling problem (Section 7). In our notation, it could be denoted by $MPS1 \mid prec \mid \sum c_k r_k(S,t)$.

### 4.2. Exact algorithms

Kelley and Walker [104] discuss both the budget problem and the deadline problem within the context of the linear time–cost tradeoff problem. In this setting (and also in a more general one, see [15]), the project cost curve $B_{\text{opt}}$ can be obtained as the inverse function of $T_{\text{opt}}$, so it suffices to consider only $B_{\text{opt}}$.

For every fixed deadline $\overline{d}$, $B_{\text{opt}}(\overline{d})$ can be expressed as a special linear program whose dual resembles a min-cost flow problem (see [74]). Using standard results from parametric linear optimization it hence follows that $B_{\text{opt}}(\overline{d})$ is a piecewise linear and convex function of the parameter $\overline{d}$.

Fulkerson [74] and Kelley [102] independently developed the same algorithm to compute the project cost curve $B_{\text{opt}}$. This algorithm uses an activity-on-arc representation of the network and iteratively calculates a sequence of less and less "cheap" cuts in the current network of critical activities by which the makespan is reduced. Every breakpoint of the project cost curve corresponds to a change of the current cut to a more expensive one.

Every such cut can be determined by a max-flow computation in which the capacities are derived from the slopes of the linear cost functions $c_j$ of the critical activities. These ideas have subsequently been improved by Phillips and Dessouky [159,160].

This algorithm is polynomial per cut ($O(|V|^2 \log|V|)$ by standard flow methods (cf. [79]), but the number of cuts to be computed may be large. Skutella [178] provides a class of examples for which the project cost curve has exponentially many breakpoints, thus requiring an exponential number of cut calculations.

The case where the possible processing times are discrete is quite common in practice. Only recently, De et al. [46] showed that, given the budget $b$, it is strongly NP-complete to decide whether there is a realization $p$ such that $c(p) \leqslant b$ and $C_{\max}(p) \leqslant 2$. This holds already for activities with at most two processing time alternatives, i.e., $|\mathcal{M}_j| \leqslant 2$.

Due to the practical importance of the problem, many (exponential time) exact algorithms have been proposed. Early examples are dynamic programming approaches by Hindelang and Muth [96] and Robinson [164], and an enumeration algorithm by Harvey and Patterson [87].

The currently best known algorithms still rely on dynamic programming, but exploit in addition the decomposition structure of the underlying network. The decomposition that facilitates the computation is known as *modular decomposition* or *substitution decomposition* and has many applications in network and other combinatorial optimization problems, see the comprehensive article by Möhring and Radermacher [129].

Its usefulness for the time–cost tradeoff problem was first observed by Frank et al. [71] and Rothfarb et al. [165] for the special case of series–parallel decompositions. The general decomposition theorem that involves arbitrary modules is due to Billstein and Radermacher [15] (see also [130]).

Because of the modular decomposition, the project cost curve needs only to be evaluated for certain indecomposable subnetworks (the factors in a composition series) of the original network. This is done by "transforming" such an indecomposable network to a series–parallel network

and then performing the "easy" calculations for the series–parallel case. The transformation into a series–parallel network successively identifies certain nodes for "duplication". Any such duplication transforms the network "closer" to a series–parallel one, but increases the computation time by a multiplicative factor.

This idea seems to be due to Robinson [164] and has been further developed by Bein et al. [13], De et al. [45] and Elmaghraby [66]. The same ideas also came up in reliability theory and seem to have influenced each other, see [6].

Demeulemeester et al. [50] provide the first implementation of this approach. They implement two strategies for finding the nodes for duplication. The first follows the theory of Bein et al. [13], which results in the minimum number of duplications required, while the second tries to minimize the number of realizations that have to be considered during the algorithm. Demeulemeester et al. [50] report on computational experience for networks with up to 45 activities without identifying a clear winner between the two strategies.

The crucial parameter in the theoretical runtime analysis of this algorithmic approach is the minimum number of node duplications needed to transform an activity-on-arc network into a series–parallel network. Bein et al. [13] refer to it as the *reduction complexity* of the network. It provides a measure for the "distance" of the given network from being series–parallel.

Such a distance measure is important for the design of polynomial-time algorithms for many network problems (see also Section 8.1 on stochastic scheduling), since computational approaches for series–parallel graphs can often be extended to algorithms for arbitrary graphs that are exponential only in the "distance" from being series–parallel, rather than in its size.

Another measure for this distance is the *factoring complexity* also introduced by Bein et al. [13], which is based on a special way of describing all paths from the source to the sink of the network. Bein et al. [13] showed that the factoring complexity provides an upper bound for the reduction complexity. Naumann [137] showed that both measures are in fact equal.

### 4.3. Approximation algorithms

The approximation behavior of the discrete time–cost tradeoff problem has recently been analyzed by Skutella [177].

He first presents a polynomial reduction to the case where every activity has at most two processing times, and one of them is zero. So $\mathcal{M}_j = \{0, \bar{p}_j\}$ or $\mathcal{M}_j = \{\bar{p}_j\}$ for activity $j \in V$.

For such a discrete time–cost tradeoff problem, Skutella defines a natural linear relaxation, which replaces $\mathcal{M}_j$ by the interval $\tilde{\mathcal{M}}_j = [0, \bar{p}_j]$ and takes as cost function $c_j$ the linear interpolation between $c_j(0)$ and $c_j(\bar{p}_j)$. All other parameters remain the same.

Now consider the deadline problem $P$ for a fixed deadline $\overline{d}$. One then first solves the linear relaxation $\tilde{P}$ for the same deadline $\overline{d}$, which yields an optimal realization $\tilde{p}$ of $\tilde{P}$ in polynomial time. Since all parameters are assumed to be integral, the obtained optimal realization $p$ will in this special case also be integral, but $\tilde{p}_j$ need not be in $\{0, \bar{p}_j\}$. This solution $\tilde{p}$ is then "rounded" to a solution of the original problem $P$ by rounding the processing time $\tilde{p}_j$ of those activities $j$ with $0 < \tilde{p}_j < \bar{p}_j$ to the lower value $p_j = 0$. Rounding to the lower value is necessary in order to preserve the deadline $\overline{d}$.

If the budget problem is considered, the rounding must be done into the other direction, i.e., $p_j = \bar{p}_j$, thus preserving the budget condition. In both cases, the produced realization $p$ of the discrete time–cost tradeoff problem is an $\ell$-approximation, where $\ell$ is the largest occurring processing time of any activity. So $C_{\max}(p) \leqslant \ell \cdot T_{\text{opt}}(b)$ and $c(p) \leqslant \ell \cdot B_{\text{opt}}(\overline{d})$, respectively.

For the deadline problem, the performance guarantee of $\ell$ cannot be improved by this rounding algorithm and it is open whether it can be improved at all. For the budget problem, however, Skutella develops better approximation algorithms. Unlike the situation for the deadline problem, he can now repair a budget violation by rounding some processing times to the higher value $\bar{p}_j$, thus "saving" part of the budget that can then be "reinvested" to shorten "critical" activities to the lower value 0.

For projects with $\bar{p}_j \in \{0, 1, 2\}$ this leads to a $\frac{3}{2}$-approximation, i.e., for a given budget $b$, the al-

gorithm produces a realization $p$ with $c(p) \leqslant b$ and $C_{\max}(p) \leqslant \lceil \frac{3}{2} T_{\text{opt}}(b) \rceil$. The running time of the algorithm is $O(|V|^3 \log |V|)$.

The NP-completeness of deciding whether a makespan of 2 can be realized with a given budget shows that the performance ratio of $\frac{3}{2}$ cannot be improved (unless $\mathscr{P} = \mathscr{NP}$).

For projects with $p_j \leqslant \ell$, Skutella uses additional partitioning techniques and obtains a strongly polynomial approximation algorithm with a performance guarantee of $2(\log_2 \ell + 1)$. A different variant yields $\frac{3}{2} \log_2 \ell + 3$.

Another idea for approximation algorithms consists in relaxing also the tight constraint (budget or deadline), thus arriving at the so-called *bicriteria approximation algorithms*. Using ideas similar to those above, Skutella shows that, for a value $0 < \mu < 1$ and an optimal time–cost pair $(\overline{d}, b)$ (i.e., $\overline{d} = T_{\text{opt}}(b)$ and $b = B_{\text{opt}}(\overline{d})$), one can in polynomial time construct a realization $p$ such that $c(p) < (1/(1-\mu))b$ and $C_{\max}(p) \leqslant (1/\mu)\overline{d}$. For $\mu = \frac{1}{2}$, this yields a realization which is at most twice as expensive and twice as long as an optimal realization for the given deadline or budget. Choosing $\mu$ uniformly at random in the interval $[1/e, 1]$ one obtains improved approximation ratios of $e/(e-1) \approx 1.58$ for the expected cost and expected makespan.

## 5. Multi-mode case

This section covers exact and heuristic algorithms for solving $MPS \mid prec \mid C_{\max}$ which is defined as follows. Like in the previous sections the project network is assumed to be acyclic and topologically sorted. Each activity $j \in V$ may be executed in one out of a set of $\mathscr{M}_j$ modes. Also, each activity may not be preempted and a mode once selected may not be changed, that is, an activity $j$ once started in mode $m \in \mathscr{M}_j$ has to be completed in mode $m$ without interruption. Processing activity $j$ in mode $m$ takes $p_{jm}$ periods and is supported by a set $\mathscr{R}^{\rho}$ of renewable and a set $\mathscr{R}^{v}$ of nonrenewable resources. Considering a horizon $T$, that is, an upper bound on the project's makespan, we have an available amount of $R_k^{\rho}$ units of renewable resource $k$ in period $t = 1, \ldots, T$ like in Section 3. The overall capacity of the nonrenewable resource $k \in \mathscr{R}^{v}$ is given by $R_k^{v}$. If activity $j$ is processed in mode $m$ then $r_{jkm}^{\rho}$ units of the renewable resource $k$ are used each period activity $j$ is in process. Similarly, activity $j$ consumes $r_{jkm}^{v}$ units of the nonrenewable resource $k$. In general, the parameters are assumed to be integer-valued. We assume the modes to be labeled with respect to nondecreasing processing times, that is, $p_{jm} \leqslant p_{j,m+1}$ for all activities $j \in V$ and modes $m \in \{1, \ldots, |\mathscr{M}_j| - 1\}$. The objective is to find a makespan-minimal schedule $\mathscr{S}$ that meets the constraints imposed by the precedence relations and by limited resource availabilities. Similar to $PS \mid prec \mid C_{\max}$ in general also $MPS \mid prec \mid C_{\max}$ is formulated mathematically in terms of a binary optimization model which makes use of binary variables $x_{jmt} = 1$, if activity $j$ is completed in mode $m$ in period $t$ (0, otherwise). For the sake of shortness, we do not present a formal model here.

If $|\mathscr{R}^{v}| \geqslant 2$ and $|\mathscr{M}_j| \geqslant 2, j \in V$, then finding a feasible solution is NP-complete (cf. [109]). However, presuming feasibility and a constant per-period availability of the renewable resources, an upper bound on the minimum makespan $T$ is given by the sum of the maximum activity processing times. Given an upper bound $T$ we can use the precedence relations and the modes of the shortest processing times to derive time windows, i.e. intervals $[EC_j, LC_j]$ similar to what has been explained in Section 3.

### 5.1. Exact algorithms

Optimal procedures for solving $MPS \mid prec \mid C_{\max}$ generalize procedures described in Section 3.1 for solving the special case $PS \mid prec \mid C_{\max}$. We summarize three algorithms.

*The Precedence Tree:* Sprecher and Drexl [186] improved the precedence tree algorithm introduced by Patterson et al. [155] by including new bounding criteria. Here, on level $g$ of the branch-and-bound tree an eligible activity $j_g$ and, subsequently, a mode $m_{j_g}$ of this activity are selected. Each combination of an eligible activity and a related mode corresponds to a descendant of the current node in the branch-and-bound tree.

*Mode and delay alternatives:* We summarize the branch-and-bound approach proposed by Sprecher et al. [187]. An eligible activity $j$ scheduled in mode $m_j$ with start time $S_j$ is said to be in process at time $t_g$ if we have $S_j \leqslant t_g < S_j + p_{jm_j}$. Eligible activities are (temporarily) started at the decision point that have already been assigned a mode at a previous level of the search tree. If there are eligible activities that have not yet been assigned a mode, that is, if $EJ_g \setminus EJ_{g-1}$ is not empty, then the set of mode alternatives is computed: A mode alternative is a mapping which assigns each activity $j \in EJ_g \setminus EJ_{g-1}$ a mode $m_j \in \mathcal{M}_j$. Selecting a mode alternative, the remaining eligible activities can be (temporarily) started at the decision point as well. Having started all eligible activities by adding them to the set $JIP_g$ of the activities in process, may have caused a resource conflict. Thus, the set of the minimal delay alternatives is computed according to the following definition: A delay alternative $\mathcal{D}\mathcal{A}_g$ is a subset of $JIP_g$ such that for each renewable resource $k \in \mathcal{R}^\rho$ it is $\sum_{j \in JIP_g \setminus \mathcal{D}\mathcal{A}_g} r^\rho_{jkm_j} \leqslant R^\rho_k$. Observe that each combination of a mode alternative and a related minimal delay alternative corresponds to a descendant of the current node in the branch-and-bound tree.

*Mode and extension alternatives:* Using again the concept of mode alternatives extension alternatives are introduced by Hartmann and Drexl [86] to construct partial schedules. More precisely, an extension alternative $\mathcal{E}\mathcal{A}_g$ is a subset of the eligible set for which $\sum_{j \in JIP_g \cup \mathcal{E}\mathcal{A}_g} r^\rho_{jkm_j} \leqslant R^\rho_k$ holds for each renewable resource $k \in \mathcal{R}^\rho$ and, moreover, $\mathcal{E}\mathcal{A}_g \neq \emptyset$ if $JIP_g = \emptyset$. At level $g$ of the branch-and-bound tree we determine the new decision point and the set of the eligible activities. Then we compute the set of mode alternatives for fixing the modes of the eligible activities that have not been eligible before, that is, those activities the modes of which have not yet been fixed. After selecting a mode alternative, compute the set of extension alternatives. Finally, select an extension alternative $\mathcal{E}\mathcal{A}_g$ and start the corresponding activities before branching to the next level. Each combination of a mode alternative and a related extension alternative corresponds to a descendant of the current node in the branch-and-bound tree.

Recently, the more general problem $MPS \mid temp \mid C_{\max}$ with general temporal constraints given by minimum and maximum start–start time lags between activities has been the subject of research in Heilmann [88], where an exact branch-and-bound procedure is presented.

A combination of the discrete time–cost tradeoff problem covered in Section 4 and of the multi-mode case dealt with in this section has been studied by Ahn and Erengüc [1].

### 5.2. Dominance rules

In Hartmann and Drexl [86] a description of several bounding rules can be found. Some of them will be revisited in what follows.

*Non-delayability rule:* If an eligible activity cannot be feasibly scheduled in any mode in the current partial schedule without exceeding its latest finish time, then no other eligible activity needs to be examined on this level.

*Local left shift rule:* If an activity that has been started at the current level of the branch-and-bound tree can be locally left shifted without changing its mode, then the current partial schedule needs not be completed.

*Multi-mode rule:* Assume that no currently unscheduled activity will be started before the finish time of a scheduled activity $j$ when the current partial schedule is completed. If a multi-mode left shift or a mode reduction of activity $j$ with resulting mode $m'_j$, $1 \leqslant m'_j \leqslant |M_j|$, can be performed on the current partial schedule and, moreover, if $r^\nu_{jkm'_j} \leqslant r^\nu_{jkm_j}$ holds for each nonrenewable resource $k$, then the current partial schedule need not be completed.

*Order swap rule:* Consider a scheduled activity the finish time of which is less than or equal to any start time that may be assigned when completing the current partial schedule. If an order swap on this activity together with any of those activities that finish at its start time can be performed, then the current partial schedule need not be completed.

*Cutset rule:* Defining a cutset of a partial schedule $\mathcal{PS}$ as the set of the activities scheduled in $\mathcal{PS}$, Sprecher and Drexl [186] proposed the following rule. Let $\overline{\mathcal{PS}}$ denote a previously eval-

uated partial schedule with cutset $CS(\overline{\mathscr{PS}})$, maximal finish time $f^{\max}(\overline{\mathscr{PS}})$ and leftover capacities $R_k^v(\overline{\mathscr{PS}})$ of the nonrenewable resources $k$. Let $\mathscr{PS}$ be the current partial schedule considered to be extended by scheduling some activity $j$ with start time $S_j$. If we have $CS(\mathscr{PS}) = CS(\overline{\mathscr{PS}})$, $S_j \geqslant f^{\max}(\overline{\mathscr{PS}})$ and $R_k^v(\mathscr{PS}) \leqslant R_k^v(\overline{PS})$ for all $k \in \mathscr{R}^v$, then $\mathscr{PS}$ needs not be completed.

*Immediate selection:* Consider an eligible activity $j$ no mode of which is simultaneously performable with any currently unscheduled activity in any mode. If the earliest feasible start time of each other eligible activity in any mode is equal to the maximal finish time of the currently scheduled activities, then $j$ is the only eligible activity that needs to be selected for being scheduled on the current level of the branch-and-bound tree.

### 5.3. Heuristic algorithms

Heuristic algorithms for solving $MPS \mid prec \mid C_{\max}$ have for instance been provided by Drexl [60], Drexl and Grünewald [61], Özdamar [149] and Kolisch and Drexl [111]. Slowiński et al. [179] address the same set of constraints, but attack the multi-criteria version of the problem. $MPSm, \sigma, \rho; 0 \mid prec \mid C_{\max}$ is the subject of Boctor [23]. While Boctor, Drexl, and Drexl and Grünewald analyze priority rule based multi-pass heuristics, Slowiński et al. provide simulated annealing algorithms, Özdamar favorizes a genetic algorithm and Kolisch and Drexl present problem specific local search algorithms.

Recently, Hartmann [84] developed the most effective and efficient heuristic algorithm for solving the general version of the problem dealt with in this section. It is a generalized version of the genetic algorithm already mentioned in Section 3.3 and basically works as follows. The genetic algorithm generates an initial population, i.e. the first generation, containing *POP* individuals and then determines their fitness values. *POP* is assumed to be an even integer. Then the population is randomly partitioned into pairs of individuals. To each pair of (parent) individuals, the crossover operator produces two new offsprings. Subsequently, the mutation operator is applied to the genotypes of the newly produced children. After computing the fitness of the offsprings, they are added to the current population, leading to a population size of $2 \cdot POP$. Then the selection operator is applied to reduce the population to its former size *POP* and to obtain the next generation to which again the crossover operator is applied. This process is repeated for a prespecified number of generations which is denoted as *GEN*.

Now a short description of the genetic operators crossover, mutation, selection is given (for details the reader is referred to Hartmann [84]).

Consider two individuals selected for *crossover*, a mother and a father. Then two random integers $w_1$ and $w_2$ with $1 \leqslant w_1, w_2 \leqslant n$ are drawn. Now two new individuals, a daughter and a son, are produced from the parents. The daughter is defined as follows: In the sequence of activities of the daughter, the positions $i = 1, \ldots, w_1$ are taken from the mother. The activity sequence of positions $i = w_1 + 1, \ldots, n$ is taken from the father. However, the activities that have already been taken from the mother may not be considered again. This definition ensures that the relative positions in the parents' activity sequences are preserved. Observe that the resulting activity sequence is precedence feasible. The modes of the activities on the positions $i = 1, \ldots, w_2$ in the daughter are defined by the mother's mode assignment. The modes of the remaining activities on the positions $i = w_2 + 1, \ldots, n$ are derived from the father's mode assignment. The son is computed similarly. However, the positions $1, \ldots, w_1$ of the son's activity sequence are taken from the father and the remaining positions are determined by the mother. Analogously, the first part up to position $w_2$ of the mode assignment of the son is taken from the father while the second part is derived from the mother. Given an activity sequence and a mode assignment for all activities an earliest start schedule is constructed.

The *mutation* is applied to each newly generated child individual and is defined as follows: Given an individual $I$ of the current population, then two random integers $q_1$ and $q_2$ with $1 \leqslant q_1 < n$ and $1 \leqslant q_2 \leqslant n$ are drawn. $q_1$ is used to modify the activity sequence by exchanging activities $j^I_{q_1}$ and $j^I_{q_1+1}$ if the result is an activity sequence which

fulfills the precedence constraints. Note that each of the changed activities keeps its assigned mode, that is, this modification does not change the mode assignment. Then a new mode for the activity on position $q_2$ is randomly chosen, that is, we redetermine $m^I(j^I_{q_2})$ by drawing a random integer out of $\{1,\ldots,\mathscr{M}_{j^I_{q_2}}\}$. While the first step may create activity sequences that could not have been produced by the crossover operator, the second step may introduce a mode that has not occurred in the current population. It should be noted that performing a mutation on an individual does not necessarily change the related schedule. This is due to the redundancy in the genetic representation.

Two variants of the *selection operator* have been considered. The first variant is a simple survival-of-the-fittest method: The original population size is restored by keeping the *POP* best individuals and removing the remaining ones from the population (ties are broken arbitrarily). The second variant is a randomized version of the survival-of-the-fittest technique.

A number of *ISL* islands are considered on which the artificial evolution as described above takes place. On each island, the evolution starts with an independently generated initial population. Let the island currently under consideration be denoted as $i$ with $1 \leqslant i < ISL$, and let the current generation be denoted as $g$ with $1 \leqslant g \leqslant GEN$. A prespecified migration probability $w_{\text{migration}}$ is used and a random number $q \in [0,1]$ is drawn to control the migration between the islands: If $q \leqslant w_{\text{migration}}$, then the fittest individual of generation $g$ leaves island $i$ and migrates to island $i+1$ where it is added to the population of generation $g$.

The stopping criterion is either to reach a prespecified number of islands as described above or, alternatively, to meet a given limit on the CPU time without bounding the number of islands. In the latter case, if *GEN* generations have been completed and the time limit has not yet been met, we skip to the next island and start a new evolution. Clearly, if the number of islands is given by *ISL*, at most $ISL \cdot POP \cdot GEN$ different individuals are calculated.

The genetic algorithm is augmented by a problem specific local search method to improve the schedule related to an individual. The approach is based on the definition of a multi-mode left shift which has been introduced by Sprecher et al. [187] in order to accelerate their branch-and-bound algorithm outlined above. A multi-mode left shift of an activity $j$ is an operation on a given schedule which reduces the finish time of activity $j$ without changing the modes or finish times of the other activities and without violating the constraints. Thereby, the mode of activity $j$ may be changed.

### 5.4. Computational results

A set of test problems constructed by the project generator ProGen which has been developed by Kolisch et al. [116] has been used. They are available in the project scheduling problem library PSPLIB. For detailed information the reader is referred to Kolisch and Sprecher [115] (cf. [114] also). The multi-mode problem sets containing instances with 10, 12, 14, and 16 nondummy activities have been used. Each of the real activities may be performed in one out of three modes. The duration of a mode varies between 1 and 10 periods. There are two renewable and two nonrenewable resources. For each problem size, a set of instances was generated by systematically varying four parameters, that is, the resource factor and the resource strength of each resource category.

In Hartmann and Drexl [86] a computational comparison of the three branching schemes in combination with bounding rules can be found. The precedence tree algorithm with the cutset rule is the fastest procedure on the average. It is two times faster than the algorithm based on mode and delay alternatives when 10 activities are considered and seven times faster for projects with 16 activities, that is, the comparison factor increases with an increasing number of activities. The algorithm based on mode and delay alternatives is at most 1.4 times faster than the algorithm based on mode and extension alternatives, hence, the latter one is outperformed by the other two algorithms with respect to average computation times. This seems to be due to the fact that branching may not be restricted to "maximal" extension alternatives. The precedence

tree algorithm is faster than the other two procedures even if the cutset rule is not included.

Noteworthy to mention that the precedence tree algorithm is more general than the other branching schemes in the sense that the case of time-varying availability profile of renewable resources can be covered.

The genetic algorithm of Hartmann [84] has been compared with the algorithm of Kolisch and Drexl [111] and Özdamar [149] on the ProGen set with 10 nondummy activities. Hartmann's algorithm produces an average deviation of 0.22% from the optimal makespan. The procedure of Kolisch and Drexl produces an average deviation of more than 0.8% from the optimal makespan. Also, the algorithm of Özdamar has an average deviation of more than 0.8%. Hence, the average deviation produced by the genetic algorithm of Hartmann [84] is nearly four times lower than those of the two heuristics from the literature.

## 6. Minimum and maximum time lags

This section is concerned with the problem $PS \mid temp \mid C_{\max}$, that is, maximum time lags between the start of different activities occur in addition to minimum ones. Maximum time lags are often needed in practice, for example, if simultaneous or nondelay execution of several activities is required, deadlines for subprojects or individual activities are prescribed, time windows for resources are given, or in scheduling of make-to-order production (cf. [138,139]).

Section 6.1 deals with modeling problem $PS \mid temp \mid C_{\max}$. Section 6.2 describes branch-and-bound methods for $PS \mid temp \mid C_{\max}$. Heuristic procedures are briefly discussed in Section 6.3. The latter two sections also summarize computational results.

### 6.1. Model

As in Section 3, $V = \{0, 1, \ldots, n, n+1\}$ is the set of activities of the project, which coincides with the node set of the corresponding activity-on-node project network. The fictitious activities 0 and $n+1$ represent the beginning and termination of the project, respectively. If there is a given minimum time lag $d_{ij}^{\min} \in \mathbb{Z}_{\geqslant 0}$ between the start of two different activities $i$ and $j$, that is, $S_j - S_i \geqslant d_{ij}^{\min}$, we introduce an arc $(i,j)$ in the project network with weight $\delta_{ij} = d_{ij}^{\min}$. If there is a given maximum time lag $d_{ij}^{\max} \in \mathbb{Z}_{\geqslant 0}$ between the start of activities $i$ and $j$, that is, $S_j - S_i \leqslant d_{ij}^{\max}$, we introduce an arc $(j,i)$ with weight $\delta_{ji} = -d_{ij}^{\max}$. The resulting network with node set $V$, arc set $E$, and arc weights $\delta_{ij}$ which satisfy the constraints $S_j - S_i \geqslant \delta_{ij}$ for $(i,j) \in E$ generally contains cycles due to maximum time lags. An appropriate specification of the minimum and maximum time lags ensures the unique assignment of the network to the underlying project (see [139]).

Given a schedule $S = (S_0, S_1, \ldots, S_{n+1})$,

$$\mathscr{A}(S,t) = \{j \in V \mid S_j \leqslant t < S_j + p_j\}$$

is the set of activities in progress at time $t \in \mathbb{Z}_{\geqslant 0}$ (or in time interval $[t, t+1[$ or period $t+1$, respectively) and

$$r_k(S,t) = \sum_{j \in \mathscr{A}(S,t)} r_{jk}$$

is the usage of renewable resource $k \in \mathscr{R}$ at time $t$. Problem $PS \mid temp \mid C_{\max}$ can then be stated as follows:

$$\min \quad S_{n+1} \tag{4}$$

$$\text{s.t.} \quad S_j - S_i \geqslant \delta_{ij} \quad (i,j) \in E$$

$$S_j \geqslant 0, \quad j \in V. \tag{5}$$

$$r_k(S,t) \leqslant R_k, \quad k \in \mathscr{R}; \quad t = 0, 1, \ldots, T-1, \tag{6}$$

where $T = \sum_{i \in V} \max(p_i, \max_{(i,j) \in E} \delta_{ij})$ is an upper bound on the minimum project duration.

It is well known that the set $\mathscr{S}_T$ of time-feasible schedules (which satisfy Eq. (5)) is nonempty exactly if the network does not contain a cycle of positive length (see [12]). The set $\mathscr{S}$ of feasible schedules (which satisfy Eqs. (5) and (6)) is generally disconnected and represents the union of convex polyhedra whose number grows exponentially in $n$. Moreover, the decision problem whether or not $\mathscr{S} \neq \emptyset$ is strongly NP-complete (cf. [12,144]).

Sometimes the constraints $S_0 = 0$ and $S_j \in \mathbb{Z}$ $(j \in V)$ are added to Eq. (5). We deleted these

constraints for the following reasons: For each optimal schedule $S$ it holds that $S_0 = 0$. Moreover, $S_0 = 0$ for all feasible schedules $S$ constructed using any heuristic method discussed in Section 6.3. Since all parameters $\delta_{ij}$ $((i,j) \in E)$ are integers, there always exists an integer optimal schedule provided that $\mathscr{S} \neq \emptyset$. All methods discussed in Sections 6.2 and 6.3 construct integer schedules.

For approximately solving $PS \mid temp \mid C_{\max}$, a decomposition approach often turns out to be expedient. A *cycle structure* of the project network is a strong component which contains at least two nodes. For each cycle structure treated as a separate subproject with original resource capacities and started at time zero, a scheduling problem corresponding to problems (4)–(6) can be stated whose (feasible) solutions are called (feasible) *subschedules*. Neumann and Zhan [140] have proven the following theorem.

**Decomposition Theorem.** *There is a feasible schedule for the project network if and only if there is a feasible subschedule for each cycle structure.*

Several heuristic procedures for (approximately) solving project scheduling problems require a *strict order* $\prec$ in node set $V$ (cf. [70]). Let $d_{ij}$ be the length of a longest path from node $i$ to node $j$ in the project network, where $d_{ij} = -\infty$ if there is no path from $i$ to $j$. For $i,j \in V, i \neq j$, we then define $i \prec j$ if and only if either (a) $d_{ij} > 0$ or (b) $d_{ij} = 0$ and $d_{ji} < 0$.

### 6.2. Branch-and-bound methods

The basic idea of branch-and-bound algorithms for solving $PS \mid temp \mid C_{\max}$ is as follows. An optimal solution to the resource relaxation of $PS \mid temp \mid C_{\max}$ (i.e. problems (4) and (5)), for example, the *earliest schedule* $ES = (ES_j)_{j \in V}$ with $ES_j = d_{0j}$, can be found in polynomial time. Starting with schedule $S = ES$, resource conflicts at points in time $t$, that is,

$$\sum_{j \in F} r_{jk} > R_k \quad \text{for some } k \in \mathscr{R} \text{ and } F \subseteq \mathscr{A}(S,t) \tag{7}$$

can be resolved successively by introducing additional temporal constraints which delay one or several activities. Set $F$ in Eq. (7) is called a *forbidden set*. If $F$ is minimal with respect to set inclusion, it is termed a *minimal forbidden set*.

Bartusch et al. [12], De Reyck [51], and De Reyck and Herroelen [54] have used "ordinary" precedence constraints of the type $S_j \geqslant S_i + p_i$, which correspond to adding arcs $(i,j)$ with weight $p_i$ to the network. Schwindt [174] has introduced *disjunctive precedence constraints*

$$S_j \geqslant \min_{i \in F \setminus \{j\}} (S_i + p_i), \tag{8}$$

where $F$ is a minimal forbidden set. Instead of delaying only one activity $j$, several activities can be delayed at the same time which form a so-called *minimal delaying alternative* (cf. [51]). Then (Eq. (8)) is replaced by

$$\min_{j \in M_2} S_j \geqslant \min_{i \in M_1} (S_i + p_i)$$

with minimal delaying alternative $M_2$ and $M_1 := \mathscr{A}(S,t) \setminus M_2$. $M_2$ is an inclusion-minimal set containing at least one element of each minimal forbidden set $F \subseteq \mathscr{A}(S,t)$.

To solve $PS \mid temp \mid C_{\max}$, Schwindt [174] has considered two partial problems. The *sequencing problem* consists of finding a set $Q$ of schedules which satisfy disjunctive precedence constraints such that $\emptyset \neq \mathscr{S}_T \cap Q \subseteq \mathscr{S}$. The corresponding *scheduling problem* consists of minimizing $S_{n+1}$ subject to $S \in \mathscr{S}_T \cap Q$. In contrast to the case of ordinary precedence constraints, the feasible region $\mathscr{S}_T \cap Q$ of the latter problem is no longer convex if disjunctive precedence constraints are used, but represents the union of convex polyhedra. A pseudopolynomial fixed-point algorithm for solving the scheduling problem has been devised by Schwindt [174].

The branch-and-bound algorithm then consists of appropriately enumerating sequencing solutions $Q_1, \ldots, Q_r$ with $\bigcup_{v=1}^{r} (\mathscr{S}_T \cap Q_v) = \mathscr{S}$ such that

$$\min_{S \in \mathscr{S}} S_{n+1} = \min_{v=1,\ldots,r} \min_{S \in \mathscr{S}_T \cap Q_v} S_{n+1}.$$

Preprocessing procedures as well as good lower bounds and fathoming rules speed up the branch-and-bound method (see [174]). An overview of recent

preprocessing techniques as well as constructive and destructive lower bounds for $PS \mid temp \mid C_{\max}$ can be found in Heilmann and Schwindt [89].

The concept of disjunctive precedence constraints markedly reduces the number of enumeration nodes of the search tree to be investigated in comparison with the branch-and-bound method by De Reyck [51]. An *experimental performance analysis* by Schwindt [174] based upon 1080 problem instances with 100 activities and five resources each (generated by the problem generator ProGen/max by Schwindt [174]) has shown that Schwindt's method solves more instances to optimality within 10 s than De Reyck's method in 100 s (using an IBM-compatible PC Pentium 200).

Another branch-and-bound procedure for $PS \mid temp \mid C_{\max}$ has been investigated by Möhring et al. [135]. The main difference lies in the way of resolving resource conflicts. Contrary to the procedures proposed by Bartusch et al. [12], De Reyck [51], De Reyck and Herroelen [54] and Schwindt [174], where additional (disjunctive) precedence constraints are introduced to resolve a conflict, the idea is to introduce ordinary *release dates* instead. That is, a resource conflict at a certain time $t$ is resolved by increasing the release dates $d_{0j}$ of activities $j \in M_2$ (i.e. the time lags between activities 0 and $j \in M_2$) according to

$$d_{0j}^{\text{new}} := \min_{i \in M_1}(S_i + p_i) \quad \text{for all } j \in M_2,$$

where $M_2$ is a minimal delaying alternative and $M_1 := \mathscr{A}(S,t) \setminus M_2$. Every node of the enumeration tree is then represented only by a vector of release dates (or start times, respectively), and, except for the values $d_{0j}$, the path lengths $d_{ij}$ $(i,j \in V)$ remain unchanged in the course of the algorithm.

On the one hand, this may in principle enlarge the enumeration tree considerably as has also been observed by Schwindt [174]. Since no precedence relation is introduced, neither "ordinary" nor disjunctive, it may happen that the same resource conflict has to be resolved several times, due to the existence of maximal time lags. But on the other hand, this way of branching gives rise to a remarkable speedup in the computation of time-feasible schedules once a branching has been performed. More precisely, the computation of optimal time-feasible schedules, and the corresponding lower bounds for newly generated nodes is then linear in the number of activities for every release date that has been increased. This is a major advantage over the procedures that introduce precedence constraints, where the complexity for the computation of time-feasible schedules is quadratic in the number of activities for every added precedence constraint (see e.g. Bartusch et al. [12]), and pseudopolynomial in the case of disjunctive precedence constraints (see the above-mentioned *scheduling problem* and [174]).

The disadvantage of multiple occurrence of the same resource conflicts, and the corresponding growth of the enumeration tree is tried to be kept small by performing *immediate selection* rules as well as a (surprisingly simple) dominance rule. Computational results indicate that, even without implementation of more sophisticated lower bounds, the procedure is competitive with the ones proposed by Schwindt [174] and De Reyck and Herroelen [54].

Recently, Dorndorf et al. [57] used constraint propagation techniques for $PS \mid temp \mid C_{\max}$. They showed that an integration of further constraints on the start times in the aforementioned sense within a new time-oriented branching scheme provides very promising results.

### 6.3. Heuristic procedures

To solve large instances of $PS \mid temp \mid C_{\max}$ approximately, truncated branch-and-bound techniques based upon Schwindt's algorithm and priority-rule methods have been developed. As to *truncated branch-and-bound procedures*, a filtered beam search technique, an $\varepsilon$-approximate algorithm, and a decomposition method have been proposed by Schwindt [174]. The decomposition method exploits the Decomposition Theorem from Section 6.1. First, for each cycle structure $\mathscr{C}$ of the network, an optimal schedule $S^{\mathscr{C}}$ is computed by the branch-and-bound algorithm. Second, each cycle structure $\mathscr{C}$ is replaced by an equivalent cycle of length zero whose arc weights are determined using $S^{\mathscr{C}}$. Third, the $\varepsilon$-approximate algorithm is applied to the resulting network.

*Priority-rule methods* for $PS \mid temp \mid C_{\max}$ have been devised and tested by Zhan [201], Neumann and Zhan [140], Brinkmann and Neumann [26], and Franck and Neumann [70] (the last reference contains the most recent results). Two different approaches have turned out to be expedient. The *sequential* or *direct method* schedules the activities one after another without considering the cycle structures of the network separately. The *contraction method* again exploits the Decomposition Theorem. First, a feasible subschedule is determined for each cycle structure. Second, each cycle structure is replaced by a single node or activity, respectively, with appropriate duration and (time-dependent) resource usage. Third, a feasible schedule for the resulting "contracted" network without cycles is computed. Fourth, a feasible schedule for the original network is determined using the schedules for the contracted network and the individual cycle structures.

To find a feasible schedule (for the whole network, the contracted network, or a cycle structure), a *serial* and a *parallel schedule generation scheme* have been developed. Among a large number of priority rules, the LST rule has turned out to be best. That is, the activity to be scheduled next is always an "eligible" activity (all of its predecessors with respect to strict order $\prec$ have already been scheduled) with smallest latest start time (cf. [70]). To take maximum time lags into account, both generation schemes contain a *backward scheduling process* which is as follows: If the earliest resource-feasible start time of the activity $j$ to be scheduled exceeds the latest possible start time of $j$ induced by some maximum time lag $d_{ij}^{\max}$, the start time of activity $i$ (and of some additional activities already scheduled) has to be enlarged appropriately.

An *experimental performance analysis* based upon 120 instances with 500 activities and five resources each (generated by ProGen/max) has provided the following main results (cf. [142,174]): The priority-rule methods are much faster than the truncated branch-and-bound procedures. Whereas the direct and contraction methods require 1 and 2 s, respectively, of computing time per instance on the average (using a PC Pentium 200), the decomposition method as slowest heuristic requires almost 1 m. The decomposition and contraction methods, which exploit the Decomposition Theorem, provide feasible (optimal) schedules for 100% (6%) and 98% (4%), respectively, of all solvable instances, where the average relative deviation of the project duration computed from the best lower bound is around 5%. The remaining heuristics solve much less instances to feasibility (the direct method only 53%), but more instances to optimality (the filtered beam search technique 62%).

## 7. Nonregular objective functions

The objective function of problem $PS \mid temp \mid C_{\max}$ discussed in Section 6 is regular, i.e. nondecreasing in the completion times of activities (in the case of a minimization problem). In this section, we deal with two kinds of nonregular objective functions where we again assume that general minimum and maximum start–start time lags are given. If the objective function to be minimized represents some measure of the variation of resource utilization, we speak of a *resource leveling problem*. In the *net present value problem*, the objective function represents the net present value of the project which is to be maximized.

### 7.1. Model

In addition to the temporal constraints (5) of $PS \mid temp \mid C_{\max}$, we explicitly require that $S_0 = 0$, $S_j \in \mathbb{Z}$ $(j \in V)$, and there is a prescribed maximum project duration $\overline{d} \in \mathbb{Z}_{\geqslant 0}$ with $\overline{d} \geqslant d_{0,n+1}$.

In the objective function of the *resource leveling problem* $PS \mid temp \mid \sum c_k f(r_k(S,t))$, $c_k > 0$ is the cost per unit of resource $k$. This problem can then be stated as follows:

$$\min \quad \sum_{k \in \mathscr{R}} c_k f(r_k(S,t)) \tag{9}$$

$$\begin{aligned} \text{s.t.} \quad & S_j - S_i \geqslant \delta_{ij} \quad (i,j) \in E, \\ & S_0 = 0, \\ & S_{n+1} \leqslant \bar{d}, \qquad (10) \\ & S_j \in \mathbb{Z}_{\geqslant 0} \quad j \in V, \\ & r_k(S,t) \leqslant R_k, \quad k \in \mathscr{R}; \quad t = 0,1,\ldots,\bar{d}-1. \end{aligned}$$

(11)

In Neumann and Zimmermann [141,142], three types of objective functions (9) are considered. If

$$f(r_k(S,t)) = \max_{t=0,1,\ldots,\overline{d}-1} r_k(S,t), \tag{12}$$

we speak of the *resource investment problem* denoted by $PS \mid temp \mid \sum c_k \max r_k(S,t)$, which is used in practice when expensive resources have to be purchased. A second type of objective function where

$$f(r_k(S,t)) = \sum_{t=0}^{\overline{d}-1} [r_k(S,t) - Y_k]^+ \tag{13}$$

measures the deviation of the consumption of resource $k$ from a target value for resource usage $Y_k \geqslant 0$. $Y_k$ may be equal to the average resource utilization $\sum_{j\in V} r_{jk}p_j/\overline{d}$. $[\ldots]^+$ in Eq. (13) can be replaced by $|\ldots|$ or $[\ldots]^2$. A third type of objective function where

$$f(r_k(S,t)) = \sum_{t=0}^{\overline{d}} [r_k(S,t) - r_k(S,t-1)]^+ \tag{14}$$

with $r_k(S,-1) = r_k(S,\overline{d}) = 0$ considers the variation of resource utilization over time and is used, for example, if the resources represent different kinds of manpower. Again $[\ldots]^+$ in Eq. (14) can be replaced by $|\ldots|$ or $[\ldots]^2$.

The basic concepts of cash flows and net present values of the cash flows of a project can be found in Russell [166] and Herroelen et al. [95]. In the objective function of the *net present value problem* $PS \mid temp \mid \sum c_j^{\mathrm{F}} \beta^{C_j}$, $\beta$ is the discount rate per period and $c_j^{\mathrm{F}}$ the cash flow associated with activity $j$, which is assumed to occur at the completion time $C_j = S_j + p_j$ of activity $j$ and can be positive (payment received) or negative (cost incurred). The net present value problem can then be formulated as follows:

$$\begin{aligned} \max \quad & \sum_{j\in V} c_j^{\mathrm{F}} \beta^{C_j} \\ \text{s.t.} \quad & (10), (11) \end{aligned}$$

### 7.2. Exact algorithms

As with problem $PS \mid temp \mid C_{\max}$, testing whether there is a feasible solution to $PS \mid temp \mid \sum c_k f(r_k(S,t))$ or $PS \mid temp \mid \sum c_j^{\mathrm{F}} \beta^{C_j}$ is strongly NP-complete.

For the *net present value problem* $PS, \infty \mid temp \mid \sum c_j^{\mathrm{F}} \beta^{C_j}$ (that is, there are no resource constraints (11)), De Reyck [51] has proposed a recursive search procedure which runs in $O(n^4)$ time. This algorithm generalizes methods for problem $PS, \infty \mid prec \mid \sum c_j^{\mathrm{F}} \beta^{C_j}$ (that is, there are only minimum time lags $d_{ij}^{\min} = p_i$) devised by Grinold [78], Elmaghraby and Herroelen [68], and Herroelen et al. [93]. De Reyck's recursive procedure starts with the earliest schedule $ES$ and tries to delay firstly activities $j$ with $c_j^{\mathrm{F}} < 0$ and secondly sets of connected activities with negative net present value as far as possible without violating the temporal constraints (10) in order to increase the net present value of the project. An *experimental performance analysis* has shown that, on the average, an instance with 100 activities can be solved in less than 1 s using a PC Pentium 60.

For the general net present value problem $PS \mid temp \mid \sum c_j^{\mathrm{F}} \beta^{C_j}$, De Reyck [51] has devised a branch-and-bound method, which is based upon De Reyck's branch-and-bound algorithm for $PS \mid temp \mid C_{\max}$, where the resource-unconstrained scheduling problems with net present value as objective function are solved by De Reyck's recursive procedure for $PS, \infty \mid temp \mid \sum c_j^{\mathrm{F}} \beta^{C_j}$. An *experimental performance analysis* with instances with up to 50 activities and five resources has shown that, in principle, the branch-and-bound method for $PS \mid temp \mid \sum c_j^{\mathrm{F}} \beta^{C_j}$ has the same effectiveness and efficiency as the corresponding procedure for $PS \mid temp \mid C_{\max}$. For $PS \mid prec \mid \sum c_j^{\mathrm{F}} \beta^{C_j}$, Icmeli and Erengüc [98] have proposed a similar branch-and-bound algorithm, which introduces additional precedence constraints to resolve resource conflicts in analogy to the branch-and-bound procedure by Demeulemeester and Herroelen [48] for $PS \mid prec \mid C_{\max}$.

For the *resource leveling problem* $PS, \infty \mid prec \mid \sum c_k f(r_k(S,t))$ with special objective functions, exact algorithms based upon enumeration, integer programming, or dynamic programming have been proposed by Ahuja [2], Easa [62], Bandelloni et al. [7], and Younis and Saad [200]. For $PS, \infty \mid temp \mid \sum c_k f(r_k(S,t))$, a time-window based branch-and-bound procedure has been

devised by Zimmermann and Engelhardt [203]. This algorithm exploits the fact that given a partial schedule $S' = (S_i)_{i \in V'}$ with $V' \subset V$, for an unscheduled activity $j \in V \setminus V'$, there is a *time window*

$$\mathscr{T}_j = \{ES_j^{S'}, ES_j^{S'} + 1, \ldots, LS_j^{S'}\}, \tag{15}$$

where

$$\begin{aligned} ES_j^{S'} &= \max\,(d_{0j}, \max_{i \in V'}(S_i + d_{ij})). \\ LS_j^{S'} &= \min\,(\overline{d} - d_{j,n+1}, \min_{i \in V'}(S_i - d_{ji})). \end{aligned} \tag{16}$$

The nodes of the enumeration tree correspond to partial schedules $S'$ with the root corresponding to $S_0 = 0$. At a node representing partial schedule $S'$, the algorithm branches as follows: Select an activity $j \in V \setminus V'$ with minimum $LS_j^{S'} - ES_j^{S'}$ and, for each $t \in \{ES_j^{S'}, \ldots, LS_j^{S'}\}$, generate a child $S^t$ by setting $S_j^t = t$. For so-called *r*-monotonous objective functions (which include functions of types (12) and (13) with $Y_k = 0$), good lower bounds at the nodes $S'$ can be computed. For a generalization of this branch-and-bound algorithm to problem $PS \mid temp \mid \sum c_k f(r_k(S,t))$ with resource constraints (11) and a preliminary performance analysis we refer to Zimmermann and Engelhardt [203].

For the *resource investment problem* $PS \mid temp \mid \sum c_k \max r_k(S,t)$, Nübel [143] has proposed a branch-and-bound procedure in analogy to the algorithm by Schwindt [174] for $PS \mid temp \mid C_{\max}$. For the nodes of the enumeration tree, which correspond to sequencing solutions, fictitious maximum resource capacities are introduced to decrease the resource capacity levels required. Resulting fictitious resource conflicts are again resolved by adding disjunctive precedence constraints. The special case $PS \mid prec \mid \sum c_k \max r_k(S,t)$ is the subject of Demeulemeester [47] and Möhring [127].

### 7.3. Heuristic procedures

For the *net present value problem* $PS \mid prec \mid c_j^F \beta^{C_j}$, several priority-rule heuristics have been proposed, for example, by Russell [167] and Padman and Smith-Daniels [152]. These papers also contain an experimental performance analysis for problem instances with 1000 or more activities and several resources. A simulated annealing approach has been presented and compared with priority-rule methods using stochastic scheduling rules by Yang et al. [199]. For the *resource leveling problem* $PS, \infty \mid prec \mid c_k f(r_k(S,t))$ with special objective functions, pseudopolynomial priority-rule methods have been devised by Burgess and Killebrew [33], Harris [82,83], Takamoto et al. [191], and Savin et al. [170]. Only small problem instances with up to 20 activities have been solved (approximately) by those methods. For problem $PS, \infty \mid temp \mid \sum c_k f(r_k(S,t))$, pseudopolynomial heuristics have been proposed and tested for instances with up to 100 activities and several resources by Brinkmann and Neumann [26].

Several variants of a *polynomial priority-rule method* have recently been presented by Zimmermann [202] and Neumann and Zimmermann [141,142], which can be applied to both $PS \mid temp \mid \sum c_k f(r_k(S,t))$ and $PS \mid temp \mid c_j^F \beta^{C_j}$. We briefly sketch the basic idea of that procedure. At first we consider the case without resource constraints (11). Given a partial schedule $S' = (S_i)_{i \in V'}, V' \subset V$, the activity to be scheduled next is either a critical activity $j \in V \setminus V'$ (that is, with slack time equal to zero) or, if there is none, an activity $j \in V \setminus V'$ with highest priority. For the net present value problem, the priority rule greatest absolute value of cash flow (GCF) is recommended. The start time $S_j$ of activity $j$ to be scheduled next equals $ES_j^{S'}$ for $c_j^F \geqslant 0$ and $LS_j^{S'}$ for $c_j^F < 0$ (where $ES_j^{S'}$ and $LS_j^{S'}$ are again given by Eq. (16)). For the resource leveling problem, the priority rules MSO (minimum number of predecessors with respect to strict order $\prec$), GRD (greatest resource demand $p_j \sum_{k \in \mathscr{R}} r_{jk}$), MST (minimum slack time), and LST (smallest latest start time) are appropriate, where the "best" rule depends on the type of objective function (cf. [141,142]). The start time $S_j$ of activity $j$ to be scheduled next is a minimizer of a penalty function which represents the additional cost arising when activity $j$ is scheduled at time $S_j$ on a certain decision set $\mathscr{D}_j$. $\mathscr{D}_j$ is a subset of the time window $\mathscr{T}_j$

(see (15)) whose cardinality is linear in $n$ and which depends on the objective function.

Two methods of generalizing the above priority-rule procedure to the case where there are resource constraints (11) are described in Neumann and Zimmermann [141,142]. An *experimental performance analysis* has shown that, on the average, an instance of problem $PS, \infty \mid temp \mid \sum c_k f(r_k(S,t))$ with 500 activities and five resources can (approximately) be solved in less than 2 s using a PC Pentium 200. For an instance of $PS \mid temp \mid \sum c_k f(r_k(S,t))$ with 200 activities and five resources, the average running time is less than 1 s. The running times for instances of the net present value problem are much smaller.

## 8. Stochastic activity durations

In real life projects, it usually does not suffice to find good schedules for fixed deterministic processing times, since these times mostly are only rough estimates and subject to unpredictable changes due to unforeseen events (weather conditions, obstruction of resource usage, delay of predecessors of an activity etc.).

In order to cope with such influences, the processing time of an activity $j$ is assumed to be a random variable $\boldsymbol{p}_j$. Then $\boldsymbol{p} = (\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n)$ denotes the (random) vector of processing times, which is distributed according to a joint probability distribution $P$. In principle, this distribution $P$ is assumed to be known (though, as will become clear later, there are methods that can deal with incomplete information about the distribution). Moreover, there may be stochastic dependencies between the different individual processing times $\boldsymbol{p}_j$, which are represented by the joint distribution $P$. In our classification, these problems are denoted by $PS \mid prec, p_j = sto \mid C_{\max}$.

The necessity of involving stochastic methods into project planning becomes obvious if one compares the "deterministic makespan" $C_{\max}(E(\boldsymbol{p}_1), \ldots, E(\boldsymbol{p}_n))$ obtained from the expected processing times $E(\boldsymbol{p}_j)$ with the expected makespan $E(C_{\max}(\boldsymbol{p}))$, even in the absence of resource constraints. There is a systematic underestimation

$$C_{\max}(E(\boldsymbol{p}_1), \ldots, E(\boldsymbol{p}_n)) \leqslant E(C_{\max}(\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n)),$$

which may become arbitrarily large with increasing number $n$ of activities or, for fixed $n$, increasing variances of the processing times (see [90]). Equality holds if and only if there is one path that is critical with probability 1. This systematic underestimation of the expected makespan has already been observed by Fulkerson [75]. The error becomes even worse if one compares the deterministic value $C_{\max}(E(\boldsymbol{p}_1), \ldots, E(\boldsymbol{p}_n))$ with quantiles $t_q$ such that $Prob\{C_{\max}(\boldsymbol{p}) \leqslant t_q\} \geqslant q$ for large values of $q$ (say $q = 0.9$ or $0.95$). This is the reason why good practical planning tools should incorporate stochastic methods. An overview about these methods is given in Section 8.1. Section 8.2 then deals with random processing times in the presence of resource constraints.

### 8.1. Stochastic scheduling without resource constraints

Due to the practical importance of stochastic scheduling, many methods have been developed over the last 35 years. For stochastically independent processing times, these methods can be roughly grouped into *simulation* methods (e.g. [34,176,180,190]), methods for *bounding or calculating the expected makespan* (e.g. [55,59,63,65,76,163]), methods for *analyzing the "most critical" path* (e.g. [181,182]), and methods for *bounding the whole distribution function of the makespan* (e.g. [56,81,107,175,183]).

An overview of the knowledge and the mathematical tools up to 1989 has been given by Möhring and Radermacher [131].

Most of these contributions have not been aware of the enormous inherent complexity of the problem, which was formally analyzed only 1988 by Hagstrom [80]. She considers the following two problems:

*MEAN:* Given a project network with discrete, independent processing times $\boldsymbol{p}_j$, compute the expected makespan $E(C_{\max}(\boldsymbol{p}))$.

*DF:* Given a project network with discrete, independent processing times $\boldsymbol{p}_j$ and a time $t$,

compute the probability $Prob\{C_{\max}(\boldsymbol{p}) \leqslant t\}$ that the project finishes by time $t$.

Hagstrom shows that the 2-state versions of these problems, in which every processing time $\boldsymbol{p}_j$ has only two discrete values, are $\#\mathscr{P}$-complete (any $\#\mathscr{P}$-complete problem is polynomially equivalent to counting the number of Hamiltonian cycles of a graph and thus in particular NP-complete). This result is derived from a fundamental result of Provan and Ball [161] on the $\#\mathscr{P}$-completeness of reliability problems and shows another connection of project scheduling to reliability theory besides time–cost tradeoff problems.

The complexity status of the general version of MEAN is open (only the 2-state version, which has a short encoding, is $\#\mathscr{P}$-complete). If the processing times $\boldsymbol{p_j}$ may take more than 2 values, the problem has a longer encoding that in principle could admit a polynomial algorithm for solving MEAN. This is, however, not the case for DF. But also for MEAN, Hagstrom provides some evidence that problems with a long encoding may still be difficult, since MEAN and DF cannot be solved in time polynomial in the number of values of $C_{\max}(\boldsymbol{p})$ unless $\mathscr{P} = \mathscr{NP}$.

These results show that efficient methods for calculating the expected makespan or quantiles of the distribution function of the makespan are very unlikely to exist, and thus (although in retrospect) justify the great interest in approximate methods such as bounds, simulation etc.

Many of the methods that provide bounds for the distribution function of the makespan transform the given network (mostly represented as an activity-on-arc network) into a series–parallel network that is more easily evaluated since series and parallel reductions of two activities $h, j$ in the network correspond to the convolution $F_h * F_j$ and pointwise product $F_h \cdot F_j$ of their processing time distribution functions $F_h, F_j$, respectively. Typical examples in this respect are the bounds by Dodin [56], Kleindorfer [107] and Spelde [183].

Möhring and Müller [128] give a unified model for such bounding results in terms of a *chain-minor* notion for project networks. A network $G_1 = (V_1, E_1)$ is a *chain-minor* of a network $G_2 = (V_2, E_2)$ if (1) and (2) below hold.

(1) Every activity $j \in V_1$ is represented in $G_2$ by a set of *copies* or *duplicates* $D(h)$, where $D(h) \cap D(j) = \emptyset$ if $h \neq j$.
(2) Every chain $C$ (the set of activities on a path) of $G_1$ is "contained" in a chain $C'$ of $G_2$ in the sense that, for every activity $j \in C$, there is a duplicate $j' \in D(j)$ with $j' \in C'$. (These duplicates $j$ may be different for different chains $C$ of $G_1$.)

Möhring and Müller [128] show that, if $G_1$ is a chain-minor of $G_2$, then one obtains a lower bound for the distribution function $F_{G_1}$ of the makespan of $G_1$ if one gives every duplicate $j'$ of an activity $j$ the same processing time distribution as activity $j$, treats them as independent, and calculates the distribution function $F_{G_2}$ of the makespan of $G_2$. In other words,

$$Prob\{C_{\max} \leqslant t \text{ in } G_1\} \geqslant Prob\{C_{\max} \leqslant t \text{ in} G_2\}$$

for every $t$.

This very general bounding principle covers the mentioned specific bounds of Kleindorfer, Spelde, Dodin and others. Moreover, if one can identify networks $G_1, G_2$ that "sandwich" the given network $G$ in the sense that $G_1$ is a chain-minor of $G$ and $G$ is a chain-minor of $G_2$, then the unknown makespan distribution function $F_G$ of $G$ is "sandwiched" by those of $G_1$ and $G_2$, i.e., $F_{G_1} \geqslant F_G \geqslant F_{G_2}$.

This brings up the question to identify networks $G_1$ and $G_2$, for which the distribution functions $F_{G_1}$ and $F_{G_2}$ are easier to evaluate. If $G_1$ and $G_2$ are chosen to be series–parallel, then the computation reduces to a sequence of convolutions and products of distribution functions. Möhring and Müller [128] show that the 2-state version of DF is still $\mathscr{NP}$-complete, but only in the weak sense. However, MEAN can in this case be solved in time polynomial in the largest number of values of the makespan of a network encountered in any series–parallel reduction sequence.

The quality of these bounds depends on the "distance" of the given network $G$ from being series–parallel, and is another motivation for studying distance measures as the reduction complexity and the factoring complexity discussed in connection with time–cost tradeoff problems in Section 4.2. In fact, any activity duplication in the

sense discussed there, leads to a lower bound for the makespan distribution function by the chain-minor result. This is the driving principle behind the bound of Dodin [56].

Another way to facilitate the calculation of the makespan distribution function is, similar to time–cost tradeoff problems, the substitution or modular decomposition. The first rigorous analysis of modular decomposition in connection with stochastic networks was done by Radermacher [162], see also [131]. It can be used both for exact calculation and for bounds.

All the methods discussed above assume that the processing time distributions are known, which usually is not the case in practice and often inhibits the use of these methods. A way to cope with this *incomplete information* is offered by the bounds of Spelde [183]. He takes as network $G_2$ in the "sandwich" a series composition of all paths of $G$, thus duplicating an activity as many times as it is contained in a path of $G$. The distribution function $F_{G_2}$ of the resulting series–parallel network $G_2$ is then the product of the distribution functions of the lengths of these paths, say $F_{G_2} = F_1 \cdot F_2 \cdot \ldots \cdot F_N$, where $F_i$ is the distribution function of the $i$th path.

If the network $G$ is large enough, i.e. all paths contain "enough" activities, then, by the central limit theorem, every $F_i$ is approximately a normal distribution function, whose mean $\mu_i$ and variance $\sigma_i^2$ are obtained as the sum of the means and variances of the processing times $\boldsymbol{p}_j$ of all activities $j$ contained in the $i$th path.

Hence it suffices to know the expected processing time $E(\boldsymbol{p}_j)$ and the variance $V(\boldsymbol{p}_j)$ of every activity in order to calculate the Spelde bound. There is, however, a complication since the number $N$ of all paths may be exponential in the size of the given network $G$. This can be overcome by calculating the first $k$ longest paths w.r.t. expected processing times $E(\boldsymbol{p}_j)$, until $Prob\{k\text{th path is longer than 1 st path}\} \leqslant \varepsilon$ for a given accuracy parameter $\varepsilon$ (say $\varepsilon = 0.05$). If $F_1, F_2, \ldots, F_k$ are the normal distribution functions of these paths, then $F_{G_2} \approx F_1 \cdot F_2 \cdot \ldots \cdot F_k$. In practice $k$ will be small.

In fact, this method contains the traditional PERT as a special case, since PERT only analyzes the distribution of the path with the longest expected path length.

Ludwig et al. [121] have implemented several of these bounds (Kleindorfer, Dodin, Spelde) and have made an extensive computational study of their bounding behavior on networks having up to 500 activities. The main conclusions from this study are that the Spelde bounds provide an excellent approximation that can be computed very fast. It usually overestimates the quantiles $t_q$ for $q \in [0.9, 1]$ only by about 5%, and thus provides a very good practical planning tool. The bounds of Dodin and Kleindorfer have an even smaller overestimation, but require complete knowledge of the processing time distributions. The accuracy of the bounds can be improved to less than 2% overestimation through the use of decomposition.

For dependent processing times, the above bounds cannot be used. Instead, there is a different approach that calculates an upper bound for the expected tardiness $E[\max\{0, C_{\max}(\boldsymbol{p}) - t\}]$ as function of $t$, which is valid for any joint distribution of the processing times and hence for all possible dependencies among them.

This approach has been investigated by Klein Haneveld [106], Meilijson and Nadas [124] and Weiss [198]. Interestingly, for discrete processing time distributions, the evaluation of this bound can be interpreted as a time–cost tradeoff problem with piecewise linear and convex cost functions that is equivalent to a linear time–cost tradeoff problem as discussed in Section 4.2. An overview of this bound is given in [131].

Another, more recent extension of stochastic network analysis concerns the combination of random processing times with the time–cost tradeoff paradigm. Here, one influences the processing time distribution of an activity by allocating more resources (money) to it. One then wants to minimize the expected makespan (or other distribution parameters) subject to a fixed budget. We refer to the work of Bowman [25] and Foldes and Soumis [69] for details.

### 8.2. Stochastic scheduling with resource constraints

We now consider random processing times together with resource constraints as for $PS \mid prec \mid C_{\max}$. This combination has often been

studied in machine scheduling, but much less in project scheduling. The model leads into the area of stochastic dynamic programming. Scheduling is done by *policies* or *strategies*. A complete characterization of all policies and subclasses thereof has been given by Möhring et al. [132,133].

For stability reasons explained there, only so-called *elementary strategies* are applicable in practice. Such a policy $\Pi$ has the following dynamic interpretation. It chooses actions at decision points. *Decision points* are $t = 0$ (project start) and activity completions. An action at time $t$ consists of starting a *feasible set* $S(t)$ at $t$, where feasible means that precedence and resource constraints are respected. The decision may of course only exploit information that has become available until the current time $t$.

In the end, when every activity has been scheduled, we have a realization $p$ of processing times and $\Pi$ has constructed a schedule $\Pi(p) = (S_1, S_2, \ldots, S_n)$ of starting times for the activities. $C^{\Pi}_{\max}(p)$ denotes the makespan of that schedule, and $E(C^{\Pi}_{\max}(\boldsymbol{p}))$ the expected makespan under policy $\Pi$. The aim then is to find a policy that minimizes the expected makespan.

Policies may be classified according to how they solve the resource conflicts. This can be modeled by looking at the set $\mathscr{F}$ of *minimal forbidden sets* $F \subseteq V$. Every proper subset $F' \subset F$ of such a set $F \in \mathscr{F}$ can in principle be scheduled simultaneously, but the set $F$ itself cannot because of the resource constraints.

A natural class of policies is the class of *preselective policies* introduced by Igelmund and Radermacher [99,100]. They solve the resource conflict on every forbidden set $F \in \mathscr{F}$ by choosing an activity $j_F \in F$ that can only start after some other activity $j \in F \setminus \{j_F\}$ has finished. This idea has also recently been used in deterministic scheduling under the name of *delaying alternatives*, see also Section 6.2.

A subclass of the class of preselective policies is obtained by letting the selected activity $j_F \in F$ always wait for the same activity $i_F \in F \setminus \{j_F\}$. Any such policy $\Pi$ can be identified with a network $G'$ constructed from the given network $G$ by adding all such precedence constraints $i_F \to j_F$. The policy $\Pi$ then constructs as $\Pi(p)$ the earliest start schedule of $G'$ for processing times $p$. These policies are therefore called ES-policies (Earliest Start policies).

Stork [189] has implemented a branch-and-bound algorithm for both classes of policies that finds an optimal preselective policy or ES-policies for stochastified ProGen instances with up to 20 activities in reasonable time. Unlike branch-and-bound algorithms for the associated deterministic setting, these algorithms require knowledge of the set $\mathscr{F}$ of minimal forbidden sets in advance and cannot use lower bounding techniques or dominance rules that involve knowledge of all processing times.

Motivated by the precedence tree concept used in branch-and-bound algorithms for $PS \mid prec \mid C_{\max}$, see Section 3.1., Möhring and Stork [134] identify an interesting subclass of the class of preselective policies, the *linear preselective policies*. Such a policy $\Pi$ chooses the waiting activities $j_F \in F$ as the last activity in $F$ according to a predefined linear ordering on the set $V$ of activities that is a topological sorting of the graph $G$ of precedence constraints.

This class of policies leads to a significant speedup in computation time, since the calculation of the expected completion time can be done more efficiently, and since many preselective policies that are dominated by others are no longer generated. Moreover, it is possible to efficiently decide whether the conflict on a currently considered forbidden set has already been indirectly settled by previous choices of waiting activities $j_F$ for other forbidden sets. These properties make it currently possible to solve most of the ProGen instances with up to 30 activities to optimality.

## 9. Further models

Enterprises are and have been facing mounting pressures to exercise reductions in costs arising from producing their goods or services and to make better use of existing staff or equipment. It is well known from practical experience that this pressure can be met, at least in part, by more efficient and intelligent planning. Successful application of these methods, however, depends to a

large degree on the ability to unambiguously and efficiently model the relevant specifics of the problems tackled. This ability, in turn, calls for expressive modeling concepts, which allow to capture a wide range of requirements appearing in real world problems. Additionally, advanced methods exploiting the degree of freedom covered by advanced models are also necessary.

Obviously, the models and methods discussed so far in this paper meet these requirements to some extent. However, there are numerous practical problem settings which require more general models. Some of them are mentioned in what follows.

(i) Recently, it has been shown by Dayanand and Padman [43,44] that the usual approach to relate to one single model which has to cover both the contractor's and the client's view of the problem might not be appropriate in practice. Consequently, in [43] models for the contractor and in [44] models for the client are discussed.

(ii) A generalization of $PS \mid prec \mid C_{\max}$ is considered in Böttcher et al. [21]. There so-called partially renewable resources are defined by assuming for each resource a capacity on subsets of periods. In [21] exact branch-and-bound and serial heuristic algorithms have been developed. The concept of partially renewable resources is a fundamental tool in order to make e.g. timetabling and shift scheduling amenable to project scheduling. In addition, partially renewable resources serve to model complicated labor regulations. Furthermore, they cover traditional renewable and nonrenewable resource constraints as special cases. Finally, in Schirmer and Drexl [171] it is shown that partially renewable resources can be used to express several kinds of logical relations between the scheduling of activities. In addition, a number of practical requirements on activities' scheduling can be formulated such as maximum or minimum quotas, as well as issues of calendarization. This underscores the expressive power of partially renewable resources.

(iii) In the multi-mode case of project scheduling all mode-activity-assignments are mutually independent in the sense that assigning a mode to one activity $j$ of a project consisting of $n$ nonpreemptable activities does not necessarily force any other activity to be processed in a specific mode. In some applications this is not feasible. Imagine, e.g., a situation in which certain activities belong together in the sense that they must be executed in the same way. This leads to the mode identity case that has been recently introduced into the project scheduling literature by Salewski et al. [168]. There it is proven that $MPS \mid prec \mid C_{\max}$ is a special case of the more general mode identity case. Moreover, it is shown that the (feasibility variant of the) mode identity case is strongly ($\mathcal{NP}$-complete) $\mathcal{NP}$-hard. Furthermore, greedy randomized adaptive search procedures are presented. Finally, it is shown that the mode identity case serves to model applications to audit-staff scheduling.

## 10. For further reading

[3,27,41,52,53,64,150,156,173,184,192,193]

## Acknowledgements

This research has been supported by Deutsche Forschungsgemeinschaft Grants Br 389/15, Dr 170/6, Mo 446/3, Ne 137/4, Pe 514/7. The authors are indebted to Alf Kimms, Frederik Stork and Marc Uetz for careful reading previous versions of the manuscript.

## Appendix A. Constraint propagation techniques

So far we have discussed methods which are tailored for solving, e.g., $PS \mid prec \mid C_{\max}$. Now, a short review of the basis of constraint propagation, the sequence consistency tests which have primarily been developed for solving the special case $PSm, 1, 1 \mid prec \mid C_{\max}$ will be given. Though designed for the special case, it is strongly conjectured that these tests are also applicable to the more general problem $PS \mid prec \mid C_{\max}$. We start with the description of some basic concepts.

### A.1. Basic concepts

The scope of inference or propagation techniques is to reach a certain level of consistency in

order to accelerate exact algorithms or local search procedures. Model based local reasoning over the constraint set makes problem specific knowledge, which is implicitly contained in the model description, explicitly available.

Most existing knowledge based scheduling systems are only capable of incorporating a small fraction of scheduling knowledge. Encouraged by this little success and the progress that is made in the development of general problem solvers in form of constraint based logic programming languages (cf. ILOG, see [120]), here we are going to restrict ourselves to the problem $PSm, 1, 1 \mid prec \mid C_{\max}$. Recognizing some typical features probably could enormously increase the power of general problem solvers in order to solve optimization, and in particular constraint satisfaction problems.

A constraint satisfaction problem (CSP) consists of a set of $n$ variables $Y_1, \ldots, Y_n$, their domains $D_1, \ldots, D_n$, respectively, and a set of constraints of these variables. An $n$-ary relation or constraint on $Y_1, \ldots, Y_n$ is a subset of the cartesian product $D_1 \times D_2 \times \cdots \times D_n$ of the domains. A solution is a value assignment of the variables such that all constraints are satisfied. As a special case a binary CSP consists only of constraints on two variables (cf. [122,125]). Obviously, project scheduling problems can be considered as constraint satisfaction problems whereby the objective function is included into the set of constraints.

A graph may serve as an illuminating representation of constraint satisfaction problems. In the dual representation each vertex of the graph corresponds to a constraint and vertices are adjacent if the vertices representing constraints have at least one variable in common. In the primal representation – the only one we are going to consider – each vertex of the graph corresponds to some variable of the CSP. An edge is a subset of the vertex set. The edge represents precisely those constraints which constitute of these and only these variables represented by the edge defining vertex set. Hence, an edge implicitly is defined by the set of all feasible tuples of variable instantiations of the edge defining constraints. The resulting graph is a hypergraph. The situation is much simpler in case of a binary CSP, i.e. a constraint $CON_{ij}(Y_i, Y_j)$ contains at most two variables $Y_i$ and $Y_j$, and corresponds to a subset of the Cartesian product $D_i \times D_j$. An edge connecting the vertices of the variables $Y_i$ and $Y_j$ corresponds to all binary relations on these two variables. The resulting graph is said to be the constraint graph of the underlying CSP. A universal relation between any two variables $Y_i$ and $Y_j$, i.e. constraints which are satisfied by all tuples of the Cartesian product $D_i$ and $D_j$, is not included into the graph. Universal constraints do not deliver any information.

Some simple consistency checks at the beginning of the search can drastically reduce the size of the search tree. These tests of consistency have the advantage that the constraint graph becomes more explicit, i.e. hidden constraints on variables currently not adjacent in the constraint graph get visible and new edges may be introduced into the graph. Hereby we say that a set of variables is $\alpha$-consistent if it is $(\alpha - 1)$-consistent and for any subset of $\alpha - 1$ variables and any instantiation of these $\alpha - 1$ variables satisfying all constraints there exists a value in the domain of the remaining variable such that all constraints on all $\alpha$ variables are satisfied. 1-consistency means that for every variable and each of its domain variables all constraints are satisfied. A set of variables is arc consistent if it is 2-consistent (this is said in relation to the constraint graph). A pair of variables $Y_i$ and $Y_j$ is path-consistent if for any feasible, i.e. $CON_{ij}(Y_i, Y_j)$ respecting, instantiation $a_i$ and $a_j$ of $Y_i$ and $Y_j$ and any sequence of edges $(CON_{i,i_1}, CON_{i_1,i_2}, \ldots, CON_{i_m,j})$ in the constraint graph there is an instantiation $a_{i_1} \in D_{i_1}, a_{i_2} \in D_{i_2}, \ldots, a_{i_m} \in D_{i_m}$ of variables $Y_{i_1}, Y_{i_2}, \ldots, Y_{i_m}$ such that all constraints $CON_{i,i_1}(a_i, a_{i_1}), \quad CON_{i_h,i_h+1}(a_{i_h}, a_{i_h+1}), CON_{i_m,j}(a_{i_m}, a_j), \ h = 1, \ldots, \ m - 1$, are satisfied. Thus path-consistency means that for any pair of variables and any explicitly feasible value pair there is also a feasible variable-value assignment on each path (edge sequence) connecting this variable pair in the constraint graph (including the universal relation). The constraint graph is arc- or path-consistent if any pair of variables is arc- or path-consistent, respectively. Arc-consistency requires an $O(e \cdot a^2)$ effort while path-consistency can be reached with an effort of $O(n^3 a^3)$ where $e$ is the number of edges in the constraint graph, $a$ is the maximum number

of elements in a domain, and $n$ is the number of variables (cf. e.g. [91]).

As mentioned earlier consistency tests yield a more explicit constraint graph. This is quite comparable to what happens during backtrack search. The enumeration only takes those constraints into account which are explicitly contained in the constraint graph, while implicitly existing edges become just visible during the search process. Finding implicitly existing constraints means to generate new knowledge in the knowledge base, it is called constraint propagation and dates back to an early idea of Waltz [197]. Clearly, the higher the level of consistency the more constraints become visible. That means, to reach arc-consistency, path- or 3-consistency can be considered as local constraint propagation. In order to reach backtrack-poor search it is necessary to make as many constraints explicit as possible, because variable instantiations which violate implicitly existing constraints usually are detected much later during the search process. Such inconsistencies then lead to backtracking. In order to reach backtrack-free search it is indispensable to make all implicitly existing constraints explicit. Montanari [136] called that the "central problem", obviously an $\mathcal{NP}$-hard one. For some special cases it is possible to require sufficient conditions in order to reach a backtrack-free or a backtrack-poor search. For instance, arc-consistency with respect to a constraint graph which is a tree guarantees backtrack-free search, cf. [72,73].

Constraint propagation, i.e. local consistency checks, can reduce the enumeration procedure before searching substantially but also during the search process it may happen that the modification of a variable domain reduces the domains of other variables, of those which are connected to the former by some constraints. That may lead to a cut of some search tree branches. Certain methods are described in the literature in order to reduce the search tree. Some of these consistency tests are described in what follows.

### A.2. Resource-based sequence consistency tests

Consider the minimum makespan problem of project scheduling with $m$ resources each of which is available in precisely one unit at a time, i.e. consider $PSm, 1, 1 \mid prec \mid C_{\max}$. The job shop scheduling problem is a special case and has received considerable attention in the literature, see the surveys by Blażewicz et al. [16,17]. There are a couple of solution approaches on job shop scheduling available which we are going to describe in the more general project scheduling setting (cf. [20,29,30,36,37,147,157,158] and the annotated bibliography by Hoogeven et al. [97]). An illuminating description of the problem is the disjunctive graph model which has been extended by Blażewicz et al. [19].

Such a disjunctive graph is a particular form of a binary constraint graph. An activity representing vertex corresponds to a variable in the constraint satisfaction model. A variable's domain consists of all possible starting times of the activity. Conjunctive arcs describe the precedence constraints and the orientation of the arc defines certain time dependencies. A disjunctive arc pair connecting two activities $i, j$, which are competing for the same resource, may be replaced by an undirected edge connecting activities $i$ and $j$, or variables $Y_i$ and $Y_j$, respectively. $Y_i$ is the domain variable which shall be instantiated with start times $S_i$. Each edge defines the constraint $CON_{ij}(Y_i, Y_j) = CON_{ji}(Y_j, Y_i)$ corresponding to $Y_i + p_i \leqslant Y_j$ or $Y_j + p_j \leqslant Y_i$, i.e. either activity $i$ is scheduled before activity $j$ or the other way round.

In order to increase efficiency a domain is represented as an interval of integers of possible processing starts, hence only the interval defining endpoints are considered as domain values. Thus an inconsistency of a tuple is not realized if one of the tuple entries belongs to the interior of an interval. Such an inconsistency is only excluded from the interval if the value of the tuple responsible for the inconsistency, will become an interval endpoint at a certain time. Consider again any constraint $CON_{ij}(Y_i, Y_j)$. Then the left bound (with respect to arc-consistency on constraint $CON_{ij}$) of the domain interval $D_i$ is defined by the temporarily earliest possible starting (release) time of activity $i$ such that there is a possible start time of activity $j$ within its domain $D_j$. The right bound (with respect to arc-consistency on constraint $CON_{ij}$) of

the domain $D_i$ is defined as the temporarily latest possible start time of activity $i$ such that there is a possible start time of activity $j$ in $D_j$ and both activities can be processed.

Assume that the order in which $i$ and $j$ are processed is not fixed. An arc-consistent domain of activity $i$ means that $D_i$ is bounded to the left by the maximum of all such earliest possible start times with respect to all activities $j$ requiring the same resource. The right bound is given by the minimum of all such latest possible start times of activity $i$ with respect to all activities $j$ requiring the same resource. Arc-consistency of an interior point of the interval is not guaranteed unless it becomes an interval endpoint at some time. Consider the disjunctive graph model. Then the longest path from the initial dummy activity to activity $i$, which defines the release date $ES_i$ of $i$, is a lower bound of the left bound of an arc-consistent interval. Correspondingly, if we subtract the tail, i.e. the length of a longest path connecting activity $i$ (including $p_i$) to the fictitious termination activity in the disjunctive graph, from the makespan (or an upper bound) then we obtain an upper bound $LS_i$ for the right endpoint of the interval. Hence, computing heads (release times) and tails generally yields only node-consistent (1-consistent) domains of possible starting points for all activities but, if one arc is selected from each disjunctive arc pair, i.e. we have an acyclic graph defining a feasible schedule, then the computation of heads and tails provides arc-consistency. Even more, without backtracking a feasible solution can be found. As the constraint graph corresponds to an activity-on-node network, these start time windows can be derived in a straightforward way. A test of path-consistency means that for any two path-consistent activities $i$ and $j$, where $i$ may be processed before $j$, and any third activity $h$ either $h,i,j$ is a possible processing sequence or $i,h,j$ or $i,j,h$. All three possibilities are checked and the intervals are modified.

Let us go into more details in order to describe a class of logical tests called sequence consistency tests which are based on resource constraints. Constraint propagation finally is a full exploration of all available resource constraints in a sense that constraints are activated to reduce variable domains unless domain reductions are no longer possible. Hence the effect of any propagation heavily depends on the kind of constraints. The efficiency depends on the constraint activation sequence, the variable and value selection in the backtrack search (if no further domain reduction can be achieved) and the level of consistency. These sequence consistency tests reduce activity domains by ruling out infeasible start time assignments. The benefit of the tests is that they can reduce the search space and direct an algorithm towards good solutions. From now on we are only interested in the tests themselves and will not address scheduling algorithms in which they can be embedded.

We assume that all activity domains have been made node-consistent (heads and tails have been calculated), i.e. we impose an upper bound on the makespan of at least one feasible schedule. First we derive the tests for the case of unit resource capacity as, for instances, encountered in machine scheduling. Finally, the results will be generalized for arbitrary usage of resources and arbitrary resource availability. A comprehensive presentation can be found in [58].

Recall $p_j$, the processing time of activity $j$, $r_{jk}$, the per period usage of renewable resource $k$, and $[ES_j, LS_j]$ ($[EC_j, LC_j]$), the start (completion) time window of activity $j$. The domain $D_j$ is the set of all possible start times $S_j$ of $j$. It is bounded by the start time window $[ES_j, LS_j]$ but some values in the start time window may be infeasible. We will use the following shorthand description w.r.t. any activity from a subset $W$ of the set $V$ of activities:

$$E(W) := \min\{ES_j \mid j \in W\},$$
$$P(W) := \sum_{j \in W} p_j$$
$$C(W) := \max\{LC_j \mid j \in W\},$$
$$LS(W) := \min\{LS_j \mid j \in W\},$$
$$EC(W) := \max\{EC_j \mid j \in W\}.$$

A sequence relation $i \to W \setminus j$ says that activity $i$ has to be scheduled (started and finished) before the start of any activity in set $W \setminus j$, i.e. there is no precedence relation between $i$ and $j$.

The idea behind all sequencing tests described now is to consider subsets $W$ of activities requiring the same resource $k$ for being processed. Within these subsets all possible activity sequences with a particular property are examined, e.g. the property that the activity sequence does not start with an activity $j$ from $W$. If all such sequences are infeasible, then we can conclude that the sequence must not have the property; therefore, we could deduce that $j$ must be first in $W$ since all sequences where this is not the case are infeasible. The sequencing tests are presented in order of decreasing strength. While a stronger condition allows a stronger conclusion, it is at the same time more likely to be inapplicable. Except for the last one all other tests in this section are derived for disjunctive scheduling problems where all activities are mutually exclusive in the sense that they exclusively occupy any required resources throughout their processing time. This is of course the case for instances of $PSm, 1, 1 \mid prec \mid C_{\max}$. However, even for instances of $PS \mid prec \mid C_{\max}$ the tests may still be used for subsets of disjunctive activities. Disjunctive tests for activity pairs lead to good lower bounds for $PS \mid prec \mid C_{\max}$, cf. Klein and Scholl [105].

Carlier and Pinson [37] have derived conditions under which it can be concluded that an activity $j$ from $W$ must be scheduled first or last in $W$, see also Carlier [35]. They even used the preemptive bound for calculation of tighter values $LS(W)$ and $EC(W)$. If an activity $j$ is scheduled before or after $W \setminus j$ we may also think of $j$ as the input or output of $W \setminus j$, which stipulates the name of the following condition.

*Input/output:* Let $j$ be an activity from a subset $W$ of the activity set $V$ where all activities of $W$ require one unit of the same resource during each period of processing. If $C(W) - E(W \setminus j) < P(W)$ then $j$ must precede all activities in $W \setminus j$. If $C(W \setminus j) - E(W) < P(W)$ then $j$ must succeed all activities in $W \setminus j$.

If the input condition holds we can reduce the domain of $j$ through an update of the latest start time $LS_j$ to $\min\{LS_j, LS(W \setminus j) - p_j\}$. Symmetrically, if the output condition holds, the earliest start time $ES_j$ can be updated to $\max\{ES_j, EC(W \setminus j)\}$. After reducing the domain of $j$ it may be possible to reduce the domains of activities in $W \setminus j$ by applying other tests. In branch-and-bound procedures that branch over disjunctive arcs, the rules may be employed to fix disjunctions, a process often called immediate selection or edge finding, cf. Brucker et al. [29].

From the input/output condition it can be deduced that an activity $j$ must be scheduled first or last in $W$. The weaker input or output condition can be used to show that a precedence relation $i \to j$ exists between an activity pair $i, j$ from $W$, cf. Blażewicz et al. [20].

*Input or output:* Let $i$ and $j$ be two activities from a subset $W$ of the activity set $V$ where all activities of $W$ require one unit of the same resource during each period of processing. If $C(W \setminus j) - E(W \setminus i) < P(W)$ then $i$ must be scheduled first or $j$ must be scheduled last in $W$. If $i$ and $j$ are two distinct activities then $i \to j$.

If the input or output condition holds and $i \neq j$ we can reduce the domains of $i$ and $j$ through an update of the latest start time $LS_i$ to $\min\{LS_i, LS_j - p_i\}$. Symmetrically, the earliest start time $ES_j$ can be updated to $\max\{ES_j, EC_i\}$. If the input or output condition holds and $i = j$ we can reduce the domain of $j$ by the interval $(LS(W \setminus j) - p_j, EC(W \setminus j))$. For the case where there are only three activities in $W$ the input or output condition allows to draw the conclusions as the $r$-set condition described in Brucker et al. [29] which present an $O(n^2)$ algorithm for checking 3-set conditions. By further relaxing the test for the input or output condition, we can still draw additional conclusions in situations where the input or output condition and the input/output condition do not hold.

*Input/output negation* (cf. [8,36,147]): Let $j$ be an activity from a subset $W$ of the activity set $V$ where all activities of $W$ require one unit of the same resource during each period of processing. If $C(W \setminus j) - ES_j < P(W)$ then $j$ must not precede all activities in $W \setminus j$. If $LC_j - E(W \setminus j) < P(W)$ then $j$ must not succeed all activities in $W \setminus j$.

The input negation extracts the interval $[0, \min\{EC_{j'} \mid j' \in W \setminus j\}[$ from the set of possible start times of $j$. Symmetrically, if the set output negation holds, $j$ must precede at least one activity

in $W$ and it extracts the interval $[\max\{LS_{j'} - p_j + 1 \mid j' \in W \setminus j\}, \infty]$ from the set of possible start times of $j$.

We now try to reason about the amount of processing time of an activity that must fall into a given interval $[t_1, t_2]$. The smallest amount of time during which an activity $j$ must be executed in the interval $[t_1, t_2]$, say the interval processing time, is:

$$p_j[t_1, t_2] := \max\{0, \min\{p_j, t_2 - t_1, EC_j - t_1, t_2 - LS_j\}\}.$$

The processing time for a set $W$ that must fall into $[t_1, t_2]$ is $P(W, [t_1, t_2]) := \sum_{j \in W} p_j[t_1, t_2]$.

*Interval pair ordering:* Let $i$ and $j$ be two activities from a subset $W$ of the activity set $V$ where all activities of $W$ require one unit of the same resource during each period of processing. If $C(\{j\}) - E(\{i\}) < P(W \setminus \{i, j\}, [E(\{i\}), C(\{j\})]) + P(\{i, j\})$ then $j \to i$.

*Interval consistency:* Let $j$ be an activity from a subset $W$ of the activity set $V$ where all activities of $W$ require one unit of the same resource during each period of processing. Let $t \in [ES_j, LC_j]$. If $t - ES_j < P(W \setminus j, [ES_j, t]) + \min\{p_j, t - ES_j\}$ then $j$ must be delayed, i.e. its earliest possible start can be increased by $P(W \setminus j, [ES_j, t])$. Symmetrically, if $LC_j - t < P(W \setminus j, [t, LC_j]) + \min\{p_j, LC_j - t\}$ then $j$ must be preferred, i.e. its latest possible completion can be decreased by $P(W \setminus j, [t, LC_j])$.

The interval consistency test can be limited to earliest start and latest completion times of activities, thus providing an efficient algorithm for this consistency test, see [58].

The aforementioned consistency tests got different names by different authors, some of them are called edge-finding. The two interval tests are also known as energetic reasoning, cf. [8]. Many of these tests generalize and extend the earlier tests on job shop scheduling, e.g. Nuijten [145] and Nuijten and Le Pape [147] update time bounds of activities using ideas presented in Carlier and Pinson [37] and incorporates the tests into a constraint satisfaction framework, cf. [146].

The results for $PSm, 1, 1 \mid prec \mid C_{\max}$ can be generalized for resources and activities with arbitrary resource availability and usage, respectively, cf. [9–11]. Assume for simplicity that the resource capacity is constant.

*Set input/output:* Let $j$ be an activity from a subset $W$ of the activity set $V$ where all activities of $W$ require at least one unit of the same resource $k$ during each period of processing.

$$\text{If } [C(W) - E(W \setminus j)]R_k < \sum_{i \in W} p_i r_{ik}, \text{ then } j \to W \setminus j.$$

$$\text{If } [C(W \setminus j) - E(W)]R_k < \sum_{i \in W} p_i r_{ik}, \text{ then } W \setminus j \to j.$$

Note that the meaning of $j \to W \setminus j$ (or $W \setminus j \to j$) is, that $j$ must start before (or end after) all activities in $W \setminus j$.

Hence, consistency checks, or roughly speaking propagation of constraints will make implicitly defined constraints more visible and will prune the search tree in a branch and bound algorithm. Obviously, $n$-consistency, where $n$ is the number of activities, immediately implies that a feasible schedule can be generated easily, however, to achieve $n$-consistency is in general not practicable. Moreover, worse upper bounds on the makespan of an optimal schedule will hardly reduce variable domains, i.e. only a few arc directions are fixed during the constraint propagation process. The better the bounds the more arc directions can be fixed, see also [38,39,123].

The main interest of propagation of constraints is the flexibility that results from the fact that each constraint propagates independently from the existence or non-existence of other constraints. It appears that the propagation process can be organized to guarantee that propagation can be done via longest path computation (cf. [19,20,39]).

## References

[1] T. Ahn, S.S. Erengüc, Resource constrained project scheduling problem with multiple crashable modes, Technical Report, College of Business Administration, University of Florida, Gainesville, USA, 1995.

[2] H.N. Ahuja, Construction Performance Control by Networks, Wiley, New York, 1976.

[3] C. Akkan, Two heuristics based on a graph induction method for the discrete time–cost tradeoff problem, Working Paper, Koç University, Istanbul, 1997.

[4] R. Alvarez-Valdes, J.M. Tamarit, Heuristic algorithms for resource-constrained project scheduling: A review and an empirical analysis, in: R. Sowiński, J. Węglarz (Eds.), Advances in Project Scheduling, Elsevier, Amsterdam, 1989, pp. 114–134.

[5] T. Baar, P. Brucker, S. Knust, Tabu-search algorithms for the resource-constrained project scheduling problem, Working Paper, Universität Osnabrück, 1997.

[6] M.O. Ball, C.J. Colbourn, J.S. Provan, Network reliability, in: M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser (Eds.), Network Models, Handbooks in Operations Research and Management Science, vol. 8, Elsevier, Amsterdam, 1995, pp. 673–762.

[7] M. Bandelloni, M. Tucci, R. Rinaldi, Optimal resource leveling using non-serial dynamic programming, European Journal of Operational Research 78 (1994) 162–177.

[8] P. Baptiste, C. Le Pape, A theoretical and experimental comparison of constraint propagation techniques for disjunctive scheduling, Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI), Montreal, Canada, 1995.

[9] P. Baptiste, C. Le Pape, W.P.M. Nuijten, Constrained-based optimization and approximation for job-shop scheduling, Proceedings of the AAAI–SIGMAN Workshop on Intelligent Manufacturing Systems, 14th International Joint Conference on Artificial Intelligence (IJCAI), Montreal, Canada, 1995.

[10] P. Baptiste, C. Le Pape, W.P.M. Nuijten, Incorporating efficient operations research algorithms in constrained-based scheduling, Proceedings of the First Joint Workshop on Artificial Intelligence and Operations Research, Timberline Lodge, Oregon, to appear.

[11] P. Baptiste, C. Le Pape, W.P.M. Nuijten, Satisfiability test and time-bound adjustments for cumulative scheduling problems, Working Paper, Université de Technologie de Compiègne, 1998.

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

[13] W.W. Bein, J. Kamburowski, M.F.M. Stallmann, Optimal reduction of two-terminal directed acyclic graphs, SIAM Journal on Computing 21 (1992) 1112–1129.

[14] C.E. Bell, J. Han, A new heuristic solution method in resource-constrained project scheduling, Naval Research Logistic 38 (1991) 315–331.

[15] N. Billstein, F.J. Radermacher, Time–cost optimization, Methods of Operations Research 27 (1977) 274–294.

[16] J. Blażewicz, W. Domschke, E. Pesch, The job shop scheduling problem: Conventional and new solution techniques, European Journal of Operational Research 93 (1996) 1–33.

[17] J. Blażewicz, K.H. Ecker, E. Pesch, G. Schmidt, J. Węglarz, Scheduling Computer and Manufacturing Processes, Springer, Berlin, 1996.

[18] J. Blażewicz, J.K. Lenstra, A.H.G. Rinnooy Kan, Scheduling subject to resource constraints, Discrete Applied Mathematics 5 (1983) 11–24.

[19] J. Blażewicz, E. Pesch, M. Sterna, A note on disjunctive graph representation, Working Paper, Universität Bonn, 1998.

[20] J. Blażewicz, E. Pesch, M. Sterna, A branch and bound algorithm for the job shop scheduling problem, in: A. Drexl, A. Kimms (Eds.), Beyond Manufacturing Resource Planning (MRP II) – Advanced Models and Methods for Production Planning, Springer, Berlin, 1998, pp. 219–244.

[21] J. Böttcher, A. Drexl, R. Kolisch, F. Salewski, Project scheduling under partially renewable resource constraints, Management Science, under review.

[22] F.F. Boctor, Some efficient multi-heuristic procedures for resource-constrained project scheduling, European Journal of Operational Research 49 (1990) 3–13.

[23] F.F. Boctor, A new and efficient heuristic for scheduling projects with resource restrictions and multiple execution modes, European Journal of Operational Research 90 (1996) 349–361.

[24] K. Bouleimen, H. Lecocq, A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem, Technical Report, Service de Robotique et Automatisation, Université de Liège, 1998.

[25] R.A. Bowman, Stochastic gradient-based time–cost tradeoffs in PERT networks using simulation, Annals of Operations Research 53 (1994) 533–551.

[26] K. Brinkmann, K. Neumann, Heuristic procedures for resource-constrained project scheduling with minimal and maximal time lags: The resource-leveling and minimum project duration problems, Journal of Decision Systems 5 (1996) 129–155.

[27] P. Brucker, Scheduling Algorithms, Springer, Berlin, 2nd edition, 1998.

[28] P. Brucker, B. Jurisch, A new lower bound for the job-shop scheduling problem, European Journal of Operational Research 64 (1993) 156–167.

[29] P. Brucker, B. Jurisch, A. Krämer, The job-shop problem and immediate selection, Annals of Operations Research 50 (1996) 73–114.

[30] P. Brucker, B. Jurisch, B. Sievers, A branch and bound algorithm for the job-shop problem, Discrete Applied Mathematics 49 (1994) 107–127.

[31] P. Brucker, S. Knust, Solving large-sized resource–constrained project scheduling problems; in: J. Węglarz (Ed.), Handbook on Recent Advances in Project Scheduling, Kluwer Academic Publishers, Dordrecht, to appear.

[32] P. Brucker, S. Knust, A. Schoo, O. Thiele, A branch and bound algorithm for the resource-constrained project scheduling problem, European Journal of Operational Research, 107 (1998) 272–288.

[33] A.R. Burgess, J.B. Killebrew, Variation in activity level on a cyclical arrow diagram, Journal of Industrial Engineering 13 (1962) 76–83.

[34] J.M. Burt, Jr., M.B. Garman, Contitional Monte Carlo: A simulation technique for stochastic network analysis, Management Science 18 (1971) 207–217.
[35] J. Carlier, The one machine sequencing problem, European Journal of Operational Research 11 (1982) 42–47.
[36] J. Carlier, E. Pinson, An algorithm for solving the job-shop problem, Management Science 35 (1989) 164–176.
[37] J. Carlier, E. Pinson, A practical use of Jackson's preemptive schedule for solving the job shop problem, Annals of Operations Research 26 (1990) 269–287.
[38] Y. Caseau, F. Laburthe, Disjunctive scheduling with task intervals, Working Paper, Ecole Normale Supérieure Paris, 1995.
[39] Y. Caseau, F. Laburthe, Cumulative scheduling with task intervals, Proceedings of the Joint Internal Conference and Symposium on Logic Programming, 1996.
[40] N. Christofides, R. Alvarez–Valdés, J.M. Tamarit, Project scheduling with resource constraints: A branch and bound approach, European Journal of Operational Research 29 (1987) 262–273.
[41] E.W. Davis, G.E. Heidorn, An algorithm for optimal project scheduling under multiple resource constraints, Management Science 17 (1971) 803–816.
[42] E.W. Davis, J.H. Patterson, A comparison of heuristic and optimum solutions in resource-constrained project scheduling, Management Science 21 (1975) 944–955.
[43] N. Dayanand, R. Padman, Payments in projects: A contractor's model, Working Paper, The Heinz School, Carnegie Mellon University, Pittsburgh, 1993.
[44] N. Dayanand, R. Padman, Project contracts and payment schedules: The client's problem, Working Paper, The Heinz School, Carnegie Mellon University, Pittsburgh, 1996.
[45] P. De, E.J. Dunne, J.B. Ghosh, C.E. Wells, The discrete time–cost tradeoff problem revisited, European Journal of Operational Research 81 (1995) 225–238.
[46] P. De, E.J. Dunne, J.B. Ghosh, C.E. Wells, Complexity of the discrete time–cost tradeoff problem for project networks, Operations Research 45 (1997) 302–306.
[47] E. Demeulemeester, Minimizing resource availability costs in time-limited project networks, Management Science 41 (1995) 1590–1598.
[48] E. Demeulemeester, W. Herroelen, A branch-and-bound procedure for the multiple resource-constrained project scheduling problem, Management Science 38 (1992) 1803–1818.
[49] E. Demeulemeester, W. Herroelen, New benchmark results for the resource-constrained project scheduling problem, Management Science 43 (1997) 1485–1492.
[50] E. Demeulemeester, W. Herroelen, S.E. Elmaghraby, Optimal procedures for the discrete time/cost trade-off problem in project networks, European Journal of Operational Research 88 (1996) 50–68.
[51] B. De Reyck, Scheduling projects with generalized precedence relations: Exact and heuristic procedures, Ph.D. Dissertation, Katholieke Universiteit Leuven, 1998.
[52] B. De Reyck, E. Demeulemeester, W. Herroelen, Local search methods for the discrete time/resource trade-off problem in project networks, Technical Report, Department of Applied Economics, Katholieke Universiteit Leuven, 1997.
[53] B. De Reyck, W. Herroelen, The multi-mode resource-constrained project scheduling problem with generalized precedence constraints, Working Paper, Katholieke Universiteit Leuven, 1997.
[54] B. De Reyck, W. Herroelen, A branch-and-bound procedure for the resource-constrained project scheduling problem with generalized precedence relations, European Journal of Operational Research, to appear.
[55] L.P. Devroye, Inequalities for the completion times of stochastic PERT networks, Mathematics of Operations Research 4 (1979) 441–447.
[56] B. Dodin, Bounding the project completion time distribution in PERT networks, Operations Research 33 (1985) 862–881.
[57] U. Dorndorf, E. Pesch, T. Phan Huy, A time-oriented branch-and-bound algorithm for project scheduling with generalized precedence constraints, Working Paper, Universität Bonn, 1998.
[58] U. Dorndorf, T. Phan Huy, E. Pesch, A survey of interval capacity consistency tests for time- and resource-constrained scheduling, in: J. Węglarz (Ed.), Handbook on Recent Advances in Project Scheduling, Kluwer Academic Publishers, Dordrecht, 1998, to appear.
[59] P.J. Downey, Distribution-free bounds on the expectation of the maximum with scheduling applications, Operations Research Letters 9 (1990) 189–201.
[60] A. Drexl, Scheduling of project networks by job assignment, Management Science 37 (1991) 1590–1602.
[61] A. Drexl, J. Grünewald, Nonpreemptive multi-mode resource-constrained project scheduling, IIE Transactions 25 (5) (1993) 74–81.
[62] S.M. Easa, Resource leveling in construction by optimization, Journal of Construction Engineering and Management 115 (1989) 302–316.
[63] S.E. Elmaghraby, On the expected duration of PERT type networks, Management Science 13 (1967) 299–306.
[64] S.E. Elmaghraby, Activity Networks – Project Planning and Control by Network Models, Wiley, New York, 1977.
[65] S.E. Elmaghraby, The estimation of some network parameters in the PERT model of activity networks: Review and critique, in: R. Slowiński, J. Węglarz, (Eds.), Advances in Project Scheduling, Elsevier, Amsterdam, 1989, pp. 371–432.
[66] S.E. Elmaghraby, Resource allocation via dynamic programming in activity networks, European Journal of Operational Research 88 (1992) 50–86.
[67] S.E. Elmaghraby, Activity nets: a guided tour through some recent developments, European Journal of Operational Research 82 (1995) 383–408.
[68] S.E. Elmaghraby, W. Herroelen, The scheduling of activities to maximize the net present value of projects,

European Journal of Operational Research 49 (1990) 35–49.
[69] S. Foldes, F. Soumis, PERT and crashing revisited: Mathematical generalizations, European Journal of Operational Research 64 (1993) 286–294.
[70] B. Franck, K. Neumann, Resource-constrained project scheduling with time windows – structural questions and priority-rule methods, Report WIOR-492, Universität Karlsruhe, 1997.
[71] H. Frank, I.T. Frisch, R. van Slyke, W.S. Chou, Optimal design of centralized computer networks, Networks 1 (1970) 43–58.
[72] E.C. Freuder, A sufficient condition of backtrack-free search, Journal of the ACM 29 (1982) 24–32.
[73] E.C. Freuder, A sufficient condition for backtrack-bounded search, Journal of the ACM 32 (1985) 755–761.
[74] D.R. Fulkerson, A network flow computation for project cost curves, Management Science 7 (1961) 167–178.
[75] D.R. Fulkerson, Expected critical path lengths in PERT networks, Operations Research 10 (1962) 808–817.
[76] W. Gaul, On stochastic analysis of project-networks, in: M.A.H. Dempster, J.K. Lenstra, A.H.G. Rinnoy Kan (Eds.), Deterministic and Stochastic Scheduling, Reidel, Dordrecht, 1982, pp. 297–309.
[77] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling theory: A survey, Annals of Discrete Mathematics 5 (1979) 287–326.
[78] R.C. Grinold, The payment scheduling problem, Naval Research Logistics Quarterly 19 (1972) 123–136.
[79] A.V. Goldberg, R.E. Tarjan, A new approach to the maximum-flow problem, Journal of the Association for Computing Machinery 35 (1988) 921–940.
[80] J.N. Hagstrom, Computational complexity of PERT problems, Networks 18 (1988) 139–147.
[81] J.N. Hagstrom, Computing the probability distribution of project duration in a pert network, Networks 20 (1990) 231–244.
[82] R.B. Harris, Precedence and Arrow Networking Techniques for Construction, Wiley, New York, 1978.
[83] R.B. Harris, Packing method for resource leveling (pack), Journal of Construction Engineering and Management 116 (1990) 39–43.
[84] S. Hartmann, Project scheduling with multiple modes: A genetic algorithm, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, no. 435, 1997.
[85] S. Hartmann, A competitive genetic algorithm for resource-constrained project scheduling, Naval Research Logistics, to appear.
[86] S. Hartmann, A. Drexl, Project scheduling with multiple modes: A comparison of exact algorithms, Networks, to appear.
[87] R.T. Harvey, J.H. Patterson, An implicit enumeration algorithm for the time/cost tradeoff problem in project network analysis, Found. Control Eng. 4 (1979) 107–117.
[88] R. Heilmann, A branch-and-bound procedure for MRCPSP/max, Report WIOR-512, Universität Karlsruhe, 1998.
[89] R. Heilmann, C. Schwindt, Lower bounds for RCPSP/max, Report WIOR-511, Universität Karlsruhe, 1997.
[90] U. Heller, On the shortest overall duration in stochastic project networks, Methods of Operations Research 42 (1981) 85–104.
[91] P. van Hentenryck, Constraint Satisfaction in Logic Programming, MIT Press, Cambridge, MA, 1989.
[92] W. Herroelen, E. Demeulemeester, B. De Reyck, Resource-constrained project scheduling – A survey of recent developments, Computers and Operations Research, to appear.
[93] W. Herroelen, E.L. Demeulemeester, P. Van Dommelen, An optimal recursive search procedure for the deterministic unconstrained max-npv project scheduling problem, Research Report 9603, Department of Applied Economics, Katholieke Universiteit Leuven, 1996.
[94] W. Herroelen, E. Demeulemeester, B. De Reyck, A classification scheme for project scheduling problems, Technical Report, Department of Applied Economics, Katholieke Universiteit Leuven, 1997.
[95] W. Herroelen, P. Van Dommelen, E. Demeulemeester, Project network models with discounted cash flows: A guided tour through recent developments, European Journal of Operational Research 100 (1997) 97–121.
[96] T.J. Hindelang, J.F. Muth, A dynamic programming algorithm for Decision CPM networks, Operations Research 27 (1979) 225–241.
[97] J.A. Hoogeven, J.K. Lenstra, S.L. Van de Velde, Sequencing and scheduling, in: M. Dell'Amico, F. Maffioli, S. Martello (Eds.), Annotated Bibliographies in Combinatorial Optimization, Wiley, New York, 1997, pp. 181–197.
[98] O. Icmeli, S.S. Erengüc, A branch and bound procedure for the resource constrained project scheduling problem with discounted cash flows, Management Science 42 (1996) 1395–1408.
[99] G. Igelmund, F.J. Radermacher, Preselective strategies for the optimization of stochastic project networks under resource constraints, Networks 13 (1983) 1–28.
[100] G. Igelmund, F.J. Radermacher, Algorithmic approaches to preselective strategies for stochastic scheduling problems, Networks 13 (1983) 29–48.
[101] T.J.R. Johnson, An algorithm for the resource-constrained project scheduling problem, Ph.D. Dissertation, MIT, Boston, USA, 1967.
[102] J.E. Kelley, Critical path planning and scheduling: Mathematical basis, Operations Research 9 (1961) 296–320.
[103] J.E. Kelley, The critical path method: Resource planning and scheduling, in: J.F. Muth, G.L. Thompson (Eds.),

Industrial Scheduling, Prentice Hall, NJ, 1963, pp. 347–365.
[104] J.E. Kelley, M.R. Walker, Critical Path Planning and Scheduling: An Introduction, Mauchly Associates, Ambler, PA, 1959.
[105] R. Klein, A. Scholl, Computing lower bounds by destructive improvement – An application to resource–constrained project scheduling, Working Paper, Technische Universität Darmstadt, 1997, European Journal of Operational Research, to appear.
[106] W.K. Klein Haneveld, Robustness against dependence in PERT: An application of duality and distributions with known marginals, Mathematical Programming Study 27 (1986) 153–182.
[107] G.B. Kleindorfer, Bounding distributions for a stochastic acyclic network, Operations Research 19 (1971) 1586–1601.
[108] U. Kohlmorgen, H. Schmeck, K. Haase, Experiences with fine-grained parallel genetic algorithms, Annals of Operations Research, to appear.
[109] R. Kolisch, Project Scheduling under Resource Constraints – Efficient Heuristics for Several Problem Classes, Physica, Heidelberg, 1995.
[110] R. Kolisch, A. Drexl, Adaptive search for solving hard project scheduling problems, Naval Research Logistics 43 (1996) 23–40.
[111] R. Kolisch, A. Drexl, Local search for nonpreemptive multi-mode resource-constrained project scheduling, IIE Transactions 29 (1997) 987–999.
[112] R. Kolisch, S. Hartmann, Heuristic algorithms for solving the resource-constrained project scheduling problem: Classification and computational analysis, in: J. Węglarz (Ed.), Handbook on Recent Advances in Project Scheduling, Kluwer Academic Publishers, Dordrecht, 1998, to appear.
[113] R. Kolisch, R. Padman, An integrated perspective of project scheduling, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, no. 463, 1997.
[114] R. Kolisch, C. Schwindt, A. Sprecher, Benchmark instances for project scheduling problems; in: J. Węglarz (Ed.): Handbook on Recent Advances in Project Scheduling, Kluwer Academic Publishers, Dordrecht, 1998, to appear.
[115] R. Kolisch, A. Sprecher, PSPLIB – A project scheduling problem library, European Journal of Operational Research 96 (1996) 205–216.
[116] R. Kolisch, A. Sprecher, A. Drexl, Characterization and generation of a general class of resource-constrained project scheduling problems, Management Science 41 (1995) 1693–1703.
[117] B. Korte, R.H. Möhring, Transitive orientation of graphs with side constraints, in: H. Noltemeier (Ed.), Proceedings of the 11th International Workshop on Graph Theoretical Concepts in Computer Science, Trauner, Linz, 1985, pp. 143–160.
[118] A. Krämer, Branch and bound methods for scheduling problems with multiprocessor tasks on dedicated processors, OR Spektrum 19 (1997) 219–227.
[119] V.J. Leon, B. Ramamoorthy, Strength and adaptability of problem-space based neighborhoods for resource-constrained scheduling, OR Spektrum 17 (1995) 173–182.
[120] C. Le Pape, Implementation of resource constraints in ILOG SCHEDULE – A library for the development of constraint-based scheduling systems, Intelligent Systems Engineering 3 (1994) 55–66.
[121] A. Ludwig, R.H. Möhring, F. Stork, A computational study on bounding the makespan distribution in stochastic project networks, Technical Report, Technische Universität Berlin, Fachbereich Mathematik, Berlin, Germany, 1998.
[122] A.K. Mackworth, Consistency in networks of relations, Artificial Intelligence 8 (1977) 99–118.
[123] P.D. Martin, D.B. Shmoys, A new approach to computing optimal schedules for the job shop scheduling problem, Proceedings of the Fifth International IPCO Conference, 1996.
[124] I. Meilijson, A. Nadas, Convex majorization with an application to the length of critical paths, Journal of Applied Probability 16 (1979) 671–677.
[125] P. Meseguer, Constraint satisfaction problems: an overview, AICOM 2 (1989) 3–17.
[126] A. Mingozzi, V. Maniezzo, S. Ricciardelli, L. Bianco, An exact algorithm for the resource-constrained project scheduling based on a new mathematical formulation, Management Science 44 (1998), 714–729.
[127] R.H. Möhring, Minimizing costs of resource requirements in project networks subject to a fixed completion time, Operations Research 32 (1984) 89–120.
[128] R.H. Möhring, R. Müller, A combinatorial approach to bound the distribution function of the makespan in stochastic project networks, Technical Report, Technische Universität Berlin, Fachbereich Mathematik, Berlin, Germany, 1998.
[129] R.H. Möhring, F.J. Radermacher, Substitution decomposition for discrete structures and connections with combinatorial optimization, Annals of Discrete Mathematics 19 (1984) 257–356.
[130] R.H. Möhring, F.J. Radermacher, The order-theoretic approach to scheduling: The deterministic case, in: R. Sowiński, J. Węglarz (eds.), Advances in Project Scheduling, Elsevier, Amsterdam, 1989, pp. 29–66.
[131] R.H. Möhring, F.J. Radermacher, The order-theoretic approach to scheduling: The stochastic case, in: R. Sowiński, J. Węglarz (Eds.), Advances in Project Scheduling, Elsevier, Amsterdam, 1989, pp. 497–531.
[132] R.H. Möhring, F.J. Radermacher, G. Weiss, Stochastic scheduling problems I – General strategies, Zeitschrift für Operations Research Ser. A 28 (1984) 193–260.
[133] R.H. Möhring, F.J. Radermacher, G. Weiss, Stochastic scheduling problems II – Set strategies, Zeitschrift für Operations Research Ser. A 29 (1985) 65–104.

[134] R.H. Möhring, F. Stork, Linear preselective policies for stochastic project scheduling, Technical Report, Technische Universität Berlin, Fachbereich Mathematik, Berlin, Germany, 1998.
[135] R.H. Möhring, F. Stork, M. Uetz, Resource constrained project scheduling with time windows: A branching scheme based on dynamic release dates, Technical Report, Technische Universität Berlin, Fachbereich Mathematik, Berlin, Germany, 1998.
[136] U. Montanari, Networks of constraints: Fundamental properties and applications to picture processing, Information Sciences 7 (1974) 95–132.
[137] V. Naumann, Measuring the distance to series-parallelity by path expressions, in: E.W. Mayr, G. Schmidt, G. Tinhofer (Eds.), Proceedings of the 20th International Workshop on Graph–Theoretic Concepts in Computer Science WG'94, Lecture Notes in Computer Science, vol. 903, Springer, Berlin, 1995, pp. 269–281.
[138] K. Neumann, C. Schwindt, Projects with minimal and maximal time lags: construction of activity-on-node networks and applications, Report WIOR-447, Universität Karlsruhe, 1995.
[139] K. Neumann, C. Schwindt, Activity-on-node networks with minimal and maximal time lags and their application to make-to-order production, OR Spektrum 19 (1997) 205–217.
[140] K. Neumann, J. Zhan, Heuristics for the minimum project-duration problem with minimal and maximal time-lags under fixed resource constraints, Journal of Intelligent Manufacturing 6 (1995) 145–154.
[141] K. Neumann, J. Zimmermann, Resource leveling for projects with schedule-dependent time windows, European Journal of Operational Research, to appear.
[142] K. Neumann, J. Zimmermann, Methods for resource-constrained project scheduling with regular and nonregular objective functions and schedule-dependent time windows; in: J. Weglarz (Ed.), Handbook on Recent Advances in Project Scheduling, Kluwer Academic Publishers, Dordrecht, 1998, to appear.
[143] H. Nübel, A branch-and-bound procedure for the resource investment problem with generalized precedence constraints, Report WIOR-516, Universität Karlsruhe, 1998.
[144] H. Nübel, C. Schwindt, A classification of shifts, schedules, and objective functions in project scheduling, Report WIOR-509, Universität Karlsruhe, 1997.
[145] W.P.M. Nuijten, Time and Resource Constrained Scheduling: A Constraint Satisfaction Approach, Ph.D. Thesis, Eindhoven University of Technology, 1994.
[146] W.P.M. Nuijten, E.H.L. Aarts, A computational study of constraint satisfaction for multiple capacitated job-shop scheduling, European Journal of Operational Research 90 (1996) 269–284.
[147] W.P.M. Nuijten, C. Le Pape, Constraint based job shop scheduling with ILOG SCHEDULER, Journal of Heuristics 3 (1998) 271–286.
[148] O. Oguz, H. Bala, A comparative study of computational procedures for the resource constrained project scheduling problem, European Journal of Operational Research 72 (1994) 406–416.
[149] L. Özdamar, A genetic algorithm approach to a general category project scheduling problem, Research Report, Marmara University, Istanbul, 1996.
[150] L. Özdamar, G. Ulusoy, A local constraint based analysis approach to project scheduling under general resource constraints, European Journal of Operational Research 79 (1994) 287–298.
[151] L. Özdamar, G. Ulusoy, A survey on the resource-constrained project scheduling problem, IIE Transactions 27 (1995) 574–586.
[152] R. Padman, D.E. Smith-Daniels, V.L. Smith-Daniels, Heuristic scheduling of resource-constrained projects with cash flows: An optimization approach, Naval Research Logistics Quarterly 44 (1997) 365–381.
[153] J.H. Patterson, Project scheduling: The effect of problem structure on heuristic performance, Naval Research Logistics Quarterly 23 (1976) 95–124.
[154] J.H. Patterson, A comparison of exact approaches for solving the multiple constrained resource project scheduling problem, Research Report, Department of Industrial and Manufacturing Systems Engineering, Lehigh University, 1984.
[155] J.H. Patterson, R. Sowiński, F.B. Talbot, J. Węglarz, An algorithm for a general class of precedence and resource constrained scheduling problems, in: R. Sowiński, J. Węglarz (Eds.), Advances in Project Scheduling, Elsevier, Amsterdam, 1989, pp. 3–28.
[156] J.H. Patterson, R. Sowiński, F.B. Talbot, J. Węglarz, Computational experience with a backtracking algorithm for solving a general class of resource constrained scheduling problems, European Journal of Operational Research 90 (1990) 68–79.
[157] E. Pesch, Learning in Automated Manufacturing, Physica, Heidelberg, 1994.
[158] E. Pesch, U. Tetzlaff, Constraint propagation based scheduling of job shops, INFORMS Journal on Computing 8 (1996) 144–157.
[159] S. Phillips, Jr., M.I. Dessouky, Solving the project time/cost tradeoff problem using the minimal cut concept, Management Science 24 (1977) 393–400.
[160] S. Phillips, Jr., M.I. Dessouky, The cut search algorithm with arc capacities and lower bounds, Management Science 25 (1979) 396–404.
[161] J.S. Provan, M.O. Ball, The complexity of counting cuts and of the probability that a graph is connected, SIAM Journal on Computing 12 (1983) 777–788.
[162] F.J. Radermacher, Invarianzaussagen für stochastische Netzpläne, Methods of Operations Research 22 (1976) 136–148.
[163] P. Robillard, M. Trahan, Expected completion time in PERT networks, Operations Research 24 (1976) 177–182.

[164] D.R. Robinson, A dynamic programming solution to the cost–time tradeoff for CPM, Management Science 22 (1975) 158–166.
[165] B. Rothfarb, H. Frank, D.M. Rosenbaum, K. Steiglitz, D.J. Kleitman, Optimal design of offshore natural-gas pipeline systems, Operations Research 18 (1970) 992–1020.
[166] A.H. Russel, Cash flows in networks, Management Science 16 (1970) 357–373.
[167] R.A. Russell, A comparison of heuristics for scheduling projects with cash flows and resource restrictions, Management Science 32 (1986) 1291–1300.
[168] F. Salewski, A. Schirmer, A. Drexl, Project scheduling under resource and mode identity constraints: Model, complexity, methods, and application, European Journal of Operational Research 102 (1997) 88–110.
[169] S.E. Sampson, E.N. Weiss, Local search techniques for the generalized resource constrained project scheduling problem, Naval Research Logistics 40 (1993) 665–675.
[170] D. Savin, S. Alkass, P. Fazio, A procedure for calculating the weight-matrix of a neuronal network for resource leveling, Advances in Engineering Software 28 (1997) 277–283.
[171] A. Schirmer, A. Drexl, Allocation of partially renewable resources – concept, models and application, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, no. 455, 1997.
[172] A. Schoo, Untere Schranken und Immediate Selection für das Resource-Constrained-Projekt Scheduling Problem, Diplomarbeit, Fachbereich Mathematik/Informatik, Unversität Osnabrück, 1996.
[173] C. Schwindt, Generation of resource-constrained scheduling problems with minimal and maximal time lags, Report WIOR-489, Universität Karlsruhe, 1996.
[174] C. Schwindt, Verfahren zur Lösung des ressourcenbeschränkten Projektdauerminimierungsproblems mit planungsabhängigen Zeitfenstern, Shaker, Aachen, 1998.
[175] A.W. Shogan, Bounding distributions for a stochastic PERT network, Networks 7 (1977) 359–381.
[176] C.E. Sigal, A.A.B. Pritsker, J.J. Solberg, The use of cutset in Monte Carlo analysis of stochastic networks, Mathematics and Computers in Simulation 21 (1979) 379–384.
[177] M. Skutella, Approximation algorithms for the discrete time–cost tradeoff problem, in: Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA, 1997, pp. 501–508.
[178] M. Skutella, Approximation and randomization in scheduling, Ph.D. Thesis, Technische Universität Berlin, Fachbereich Mathematik, Berlin, Germany, 1998.
[179] R. Sowiński, B. Sońiewicki, J. Węglarz, DSS for multiobjective project scheduling subject to multiple-category resource constraints, European Journal of Operational Research 79 (1994) 220–229.
[180] R.M. van Slyke, Monte Carlo methods and the PERT problem, Operations Research 11 (1963) 839–860.
[181] H. Soroush, Risk taking in stochastic PERT networks, European Journal of Operational Research 67 (1993) 221–241.
[182] H.M. Soroush, The most critical path in a PERT network, Journal of the Operational Research Society 45 (1994) 287–300.
[183] H.G. Spelde, Stochastische Netzpläne und ihre Anwendung im Baubetrieb, Ph.D. Thesis, Rheinisch-Westfälische Technische Hochschule Aachen, 1976.
[184] A. Sprecher, Resource-Constrained Project Scheduling – Exact Methods for the Multi-Mode Case, Springer, Berlin, 1994.
[185] A. Sprecher, Solving the RCPSP efficiently at modest memory requirements, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, no. 426, 1996.
[186] A. Sprecher, A. Drexl, Solving multi-mode resource-constrained project scheduling problems by a simple, general and powerful sequencing algorithm, European Journal of Operational Research, 107 (1998) 431–450.
[187] A. Sprecher, S. Hartmann, A. Drexl, An exact algorithm for project scheduling with multiple modes, OR Spektrum 19 (1997) 195–203.
[188] J.P. Stinson, E.W. Davis, B.M. Khumawala, Multiple resource-constrained scheduling using branch and bound, AIIE Transactions 10 (1978) 252–259.
[189] F. Stork, A branch and bound algorithm for minimizing expected makespan in stochastic project networks with resource constraints, Technical Report, Technische Universität Berlin, Fachbereich Mathematik, Berlin, Germany, 1998.
[190] R.S. Sullivan, J.C. Hayya, A comparison of the method of bounding distributions (MBD) and Monte Carlo simulation for analyzing stochastic acyclic networks, Operations Research 28 (1980) 614–617.
[191] M. Takamoto, N. Yamada, Y. Kobayashi, H. Nonaka, Zero–one quadratic programming algorithm for resource leveling of manufacturing process schedules, Systems and Computers in Japan 26 (1995) 68–76.
[192] F.B. Talbot, Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case, Management Science 28 (1982) 1197–1210.
[193] F.B. Talbot, J.H. Patterson, An efficient integer programming algorithm with network cuts for solving resource-constrained scheduling problems, Management Science 24 (1978) 1163–1174.
[194] A. Thesen, Heuristic scheduling of activities under resource and precedence restrictions, Management Science 23 (1976) 412–422.
[195] E. Tsang, Foundations of Constraint Satisfaction, Academic Press, London, 1993.
[196] G. Ulusoy, L. Özdamar, A constrained-based perspective in resource constrained project scheduling, International Journal of Production Research 32 (1994) 693–705.
[197] D. Waltz, Understanding line drawings of scenes with shadows, in: P.H. Winston (Ed.), Psychology of Com-

puter Vision, McGraw Hill, Cambridge, MA, 1975, pp. 19–91.

[198]z G. Weiss, Stochastic bounds on distributions of optimal value functions with applications to PERT, network flows and reliability, Operations Research 34 (1986) 595–605.

[199] K.K. Yang, L.C. Tay, C.C. Sum, A comparison of stochastic scheduling rules for maximizing project net present value, European Journal of Operational Research 85 (1995) 327–339.

[200] M.A. Younis, B. Saad, Optimal resource leveling of multi-resource projects, Computers and Industrial Engineering 31 (1996) 1–4.

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

[202] J. Zimmermann, Heuristics for resource-leveling problems in project scheduling with minimum and maximum time lags, Report WIOR–491, Universität Karlsruhe, 1997.

[203] J. Zimmermann, H. Engelhardt, Lower bounds and exact algorithms for resource levelling problems, Report WIOR-517, Universität Karlsruhe, 1998.