Section III

Quantitative Models, Data Structuring and Information Processing

SCHEDULING PROJECT NETWORKS WITH RESOURCE CONSTRAINTS AND TIME WINDOWS

M. BARTUSCH [1], R.H. MÖHRING [2] and F.J. RADERMACHER [3]

[1] *Lehrstuhl für Informatik und Operations Research, Universität Passau, W.-Germany.*
[2] *Fachbereich Mathematik, TU Berlin W.-Germany,*
supported by Sonderforschungsbereich 303 (DFG), Universität Bonn
[3] *Forschungsinstitut für Anwendungsorientierte Wissensverarbeitung, Ulm, W.-Germany*

Abstract

Project networks with time windows are generalizations of the well-known CPM and MPM networks that allow for the introduction of arbitrary minimal and maximal time lags between the starting and completion times of any pair of activities.

We consider the problem to schedule such networks subject to arbitrary (even time dependent) resource constraints in order to minimize an arbitrary regular performance measure (i.e. a non-decreasing function of the vector of completion times). This problem arises in many standard industrial construction or production processes and is therefore particularly suited as a background model in general purpose decision support systems.

The treatment is done by a structural approach that involves a generalization of both the disjunctive graph method in job shop scheduling [1] and the order theoretic methods for precedence constrained scheduling [18,23,24]. Besides theoretical insights into the problem structure, this approach also leads to rather powerful branch-and-bound algorithms. Computational experience with this algorithm is reported.

Keywords: Scheduling, project networks, MPM-networks, time-windows, order theoretic approach to scheduling, disjunctive graph method.

1. Introduction

Real-life scheduling problems in industrial applications typically involve the optimization of a complicated (usually non-linear) multiattribute performance measure subject to different kinds of constraints such as technological precedence, time lags or windows, time-varying resource requirements and resource availabilities, etc. This constitutes a situation particularly suited for the development of general purpose decision support systems. The usual scheduling models in machine scheduling [9] or project scheduling [10,23] are either too special to model such a situation or just reduce the problem to a (mixed) integer programming problem [31,22] and apply general integer programming techniques. This paper aims at closing the gap between practical needs and theoretical tools by

investigating a very general class of deterministic scheduling problems which allows for:

- arbitrary precedence constraints involving *schedule-dependent time windows*, i.e. minimal and maximal time lags between starting times and completion times of any two activities,
- different resource types, whose availability may change (in discrete jumps) over time,
- resource requirements per activity involving several types and amounts that may vary (in discrete jumps) with the processing of each activity,
- cost criteria (performance measures) that are arbitrary non-decreasing functions of the activities' completion times, and thus may involve both "real cost" and "individual preferences".

The first important step in one analysis of such problems is the development of a *suitable theoretical model* that represents the *basic features* of such scheduling problems by relatively *simple mathematical structures* that allow both *theoretical investigations* (such as representation theorems for the set of feasible solutions and the optimum value function, sensitivity analysis etc.) and are suited for a "good" *algorithmic treatment* (development and use of simple standard data structures, separation of subtasks with high complexity etc.).

This is achieved in sect. 2 by a far-reaching reduction to the well-investigated order-theoretic approach for project scheduling with partial orders as precedence constraints [18,20,23,24].

Instead of partial orders, precedence constraints and time windows are now modeled by a "distance matrix" that represents a minimum potential in a directed graph. This technique is closely related to the MPM-method for the temporal analysis of project networks [27,28,10].

Furthermore, the appropriate use of time windows allows for a reduction of the *time-dependent* resource requirements and availabilities to *constant*, i.e. time independent, requirements and availabilities. These can, as in the order-theoretic approach, be modeled by a system of *forbidden sets*, i.e. sets of networks activities that may never be scheduled simultaneously. This reduction is important, as it opens the way for a *structural approach* which is close to that for the order-theoretic case.

This approach is presented in sect. 3. We first show that if a system of temporal constraints in the form of time windows has a feasible solution, then there is an "embedded" partial order whose chains represent precedence constraints and whose antichains represent all principal possibilities for simultaneous scheduling of jobs some of which are "forbidden" by the system of forbidden sets. As in the order-theoretic case, construction of feasible schedules then means introducing additional precedence constraints on these forbidden sets. In the general case treated here, however, not all precedence constraints are possible, as they may conflict with some time window constraint.

This establishes one of the major differences with the order-theoretic case. It has several far-reaching consequences. In theoretical respect, it implies e.g. that testing for the existence of a feasible solution is already NP-hard (theorem 3.10), and that the stability behaviour w.r.t. variation of processing times is much more restricted (remark 3.23). Section 3 will give a rigorous analysis of these aspects.

Section 4 presents a branch and bound algorithm based on the structural approach of sect. 3. It generalizes both the disjunctive graph method for job shop scheduling problems [1] and the branch and bound method for the order-theoretic case [23,24]. An important speed-up is obtained by reduction techniques that reduce the dimension of the problem to the number of jobs that actually require scarce resources (are in some forbidden set) or whose completion time is essential in the cost function. This reduction to the so-called *essential jobs* reduces the computational effort per node of the underlying branching tree considerably. A further reduction is obtained by time windows that restrict the choice of precedence constraints on a forbidden set. Though theoretically harder to deal with, with respect to the branch and bound algorithm they may prune branches of the tree and thus restrict the search for good solutions. We close with demonstrating our computational experience with this algorithm.

2. The general model

2.1. JOBS, SCHEDULES, AND PERFORMANCE MEASURES

The basic entities of the scheduling problems considered are the *activities* or *jobs*. The set of all activities is denoted by $A = \{\alpha_1, \ldots, \alpha_n\}$, individual activities are denoted by α_i $(i = 1, \ldots, n)$ or α, β, γ, etc.

These jobs represent units of the scheduling problem that must be scheduled *without preemption*. (With respect to preemptive scheduling models this means that preemptions can only take place at specified moments.) As will be seen later, it is convenient to permit here also *artificial jobs* in order to represent cost terms (milestones) or resource contraints in a simple way. Each job α_i (real or artificial) has a fixed *processing time* $x_i = x(\alpha_i) > 0$. The vector $x = (x_1, \ldots, x_n)$ denotes the joint vector of processing times.

A *schedule* is an assignment of starting times to the jobs $\alpha_1, \ldots, \alpha_n$, i.e. a vector $S = (S_1, \ldots, S_n)$, where $S_i = S(\alpha_i)$ denotes the *starting time* of job α_i, i.e. the time at which processing of job α_i starts. The time at which job α_i has been completely processed is called the *completion time* of α_i and is denoted by $C_i = C(\alpha_i)$. Since we assume that processing times are deterministic and as preemptions are not permitted, the completion times are in our model uniquely determined by

$$S_i + x_i = C_i. \tag{2.1}$$

The vector $C = (C_1, \ldots, C_n)$ of completion times is in nearly all scheduling models regarded as the list of parameters or attributes that determines the

"goodness" of a schedule. This is usually [26,15] done by transforming the multi-dimensional attribute $C=(C_1,\ldots,C_n)$ by a function κ: $\mathbb{R}^n_{\geqslant} \to \mathbb{R}^1$ onto a one-dimensional scale representing cost or utility.

So given κ and a schedule S, $\kappa(S;\ x)=\kappa(C_1,\ldots,C_n)=\kappa(S+x)$ denotes the *cost* resulting from scheduling jobs $\alpha_1,\ldots,\alpha_n$ according to κ. The only assumption we make on κ is that it is non-decreasing w.r.t. the componentwise ordering of $\mathbb{R}^n$, i.e.

$$(2.2) \qquad (C_1,\ldots,C_n) \leqslant (C'_1,\ldots,C'_n) \Rightarrow \kappa(C_1,\ldots,C_n) \leqslant \kappa(C'_1,\ldots,C'_n).$$

These so-called *regular measures of performance* [8,26] or *regular cost functions* cover the standard cost functions in machine scheduling such as makespan, (weighted) flowtime, tardiness costs etc, and are general enough to allow also for preference terms, real cost terms, milestones and many other cost terms occuring in industrial applications.

2.2. TEMPORAL CONSTRAINTS

Schedules are subject to two types of constraints, *temporal constraints* and *resource constraints*. In their most general form (which is standard e.g. in applications in building industrie [13,30]), the *temporal constraints* are given by arbitrary minimum and/or maximum time lags between starting times and/or completion times of any two jobs. For instance, time lags between starting times of jobs α_i and α_j have the form

$$(2.3) \qquad S_i + l_{ij}^{\min} \leqslant S_j \leqslant S_i + l_{ij}^{\max}.$$

$l_{ij}^{\min}$ and $l_{ij}^{\max}$ are called the *minimum* and *maximum start-to-start lag* of job α_j relative to job α_i, respectively, and $W_{ij} := [S_i + l_{ij}^{\min},\ S_i + l_{ij}^{\max}]$ the *time window* of S_j relative to S_i. *Start-to-finish* lags, *finish-to-start* lags, *finish-to-finish* lags, and the associated windows are defined analogously. A schedule $S=(S_1,\ldots,S_n)$ is called *time-feasible*, if the starting times S_i fulfil all inequalities given by the time lags and (2.1). $\mathscr{S}_T \subseteq \mathbb{R}^n_{\geqslant}$ denotes the set of all time-feasible schedules. $\mathscr{S}_T$ may be empty since the constraints may be contradictory.

These temporal constraints contain the more common partial order precedence constraints as a special case by considering only start-to-start time windows of the form $W_{ij}=[S_i+x_i,\ \infty]$. Also release times and due dates can easily be represented in this model.

Graphical representations of networks with time lags usually draw for each job a small rectangle whose left (right) side denotes its start (completion). Time lags are represented by arrows between the associated sides of the rectangles. For instance, in the example network in fig. 1, the arrow from job 3 to job 6 represents the finish-to-finish time lags $l_{36}^{\min}=3$ and $l_{36}^{\max}=10$.

Since we are dealing with fixed, deterministic processing times and time lags, all different kinds of time lags may be represented in a *standardized form* by

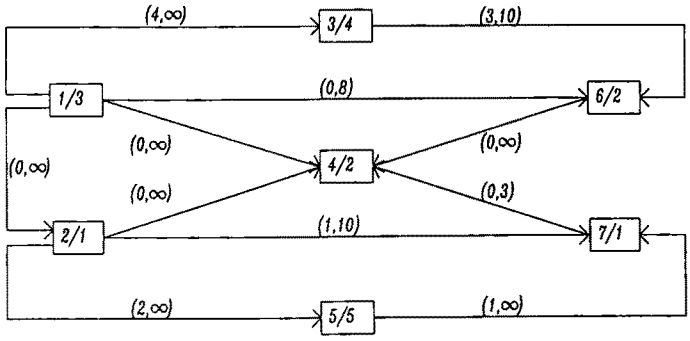

Fig. 1. An example network with time lags.

reducing them to just one type; e.g. to minimum start-to-start lags. This is achieved by replacing completion times C_i according to (2.1) by $S_i + x_i$ and by transforming each inequality into the form $S_i + l_{ij} \leqslant S_j$ (by allowing l_{ij} to become negative). Alltogether, one obtains the following *transformation rules*:
(2.4) start-to-start lags:

$$S_i + l_{ij}^{\min} \leqslant S_j \rightarrow S_i + l_{ij} \leqslant S_j \quad \text{with } l_{ij} = l_{ij}^{\min}$$

$$S_i + l_{ij}^{\min} \geqslant S_j \rightarrow S_j + l_{ji} \leqslant S_i \quad \text{with } l_{ji} = -l_{ij}^{\max}$$

(2.5) start-to-finish lags:

$$S_i + l_{ij}^{\min} \leqslant C_j \rightarrow S_i + l_{ij} \leqslant S_j \quad \text{with } l_{ij} = l_{ij}^{\min} - x_j$$

$$S_i + l_{ij}^{\max} \geqslant C_j \rightarrow S_j + l_{ji} \leqslant S_i \quad \text{with } l_{ji} = x_j - l_{ij}^{\max}$$

(2.6) finish-to-start lags:

$$C_i + l_{ij}^{\min} \leqslant S_j \rightarrow S_i + l_{ij} \leqslant S_j \quad \text{with } l_{ij} = x_i + l_{ij}^{\min}$$

$$C_i + l_{ij}^{\max} \geqslant S_j \rightarrow S_j + l_{ji} \leqslant S_i \quad \text{with } l_{ji} = -x_i - l_{ij}^{\max}$$

(2.7) finish-to-finish lags:

$$C_i + l_{ij}^{\min} \leqslant C_j \rightarrow S_i + l_{ij} \leqslant S_j \quad \text{with } l_{ij} = x_i - x_j + l_{ij}^{\min}$$

$$C_i + l_{ij}^{\max} \geqslant C_j \rightarrow S_j + l_{ji} \leqslant S_i \quad \text{with } l_{ji} = x_j - x_i - l_{ij}^{\max}$$

This reduction permits the representation of the temporal constraints by a digraph $G = (V, E)$ with edge weights as follows: G has a *vertex* for each job (i.e. $V = A$), and an *edge* (α_i, α_j) directed from α_i to α_j if there is a constraint of the form $S_i + l_{ij} \leqslant S_j$ with $l_{ij} > -\infty$. In that case, the maximum value l_{ij} of all these constraints is assigned as *weight* or *length* l_{ij} to the edge (α_i, α_j). G is called the *digraph of the temporal constraints* or simply the *constraint digraph*. For the example of fig. 1, the constraint digraph is given in fig. 2.

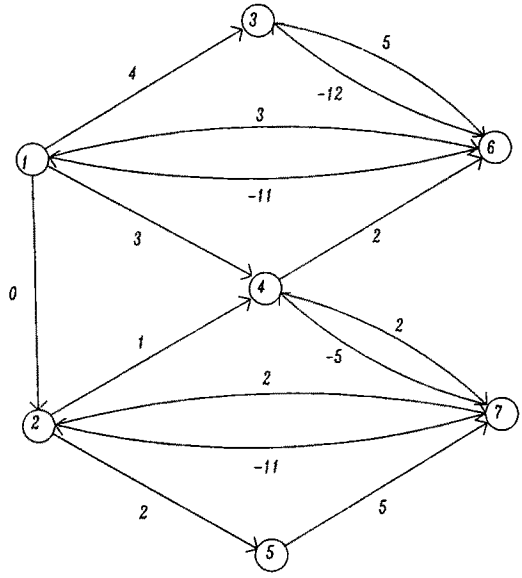

Fig. 2. The digraph associated with the example of fig. 1.

Obviously, a time-feasible schedule $S = (S_1, \ldots, S_n)$ then corresponds to an assignment of numbers π_i to the vertices $\alpha_i (i = 1, \ldots, n)$ such that

$$(2.8) \quad \begin{cases} a) & \pi_i \geqslant 0 \\ b) & \pi_i + l_{ij} \leqslant \pi_j \text{ for each edge } (\alpha_i, \alpha_j) \text{ of } G \end{cases}$$

holds, and vice versa.

Numbers π_i $(i = 1, \ldots, n)$ fulfilling (2.8)b) are known as (*node*) *potentials* in graph theory, and there is a well-developped theory about them [5,27,28]. This theory is also the basis of the so-called *metrapotential-method* (MPM) for project networks [27,28,21], which deals with start-to-start lags, and thus, by the above reduction, essentially also covers the general temporal constraints of our model. Further applications to job-shop scheduling are given in [6]. The main results applying here are formulated in the following proposition.

2.1. PROPOSITION

Let $G = (V, E)$ be a digraph with vertex set $V = \{\alpha_1, \ldots, \alpha_n\}$, edge set $E \subseteq V \times V$ and edge lengths $l_{ij} \in \mathbb{R}$ [1] *for each edge $(\alpha_i, \alpha_j) \in E$.*

(1) *There exists a potential for G iff G has no directed circuit of positive length.* [1]

(2) *The set $\mathcal{P}$ of all potentials for G is a convex polyhedron in $\mathbb{R}^n$ which is either empty or unbounded.*

[1] For all graphtheoretic notions not defined here, see [14].

(3) *If* $\mathscr{P} \neq \emptyset$, *then* $\mathscr{P} \cap \mathbb{R}^n_{\geqslant}$ (*i.e. the set of solutions of* (2.8)) *contains a unique potential* $\pi^0 = (\pi_1^0, \ldots, \pi_n^0)$ *that is componentwise smaller than all other* $\pi \in \mathscr{P} \cap \mathbb{R}^n_{\geqslant}$.

Sketch of proof: Let α_i be a vertex on a directed circuit with total length l. The edge conditions (2.8) b) along the circuit then imply that $\pi_i + l \leqslant \pi_i$ which is only possible if $l \leqslant 0$.

To show the other direction of (1) assume w.l.o.g. that there is a vertex (α_1, say) from which each other vertex can be reached by a directed path with non-negative arc lengths. (Otherwise introduce a new vertex with an arc of length 0 to every other vertex.) Put $\lambda_1 = 0$ and let, for $i = 2, \ldots, n$, λ_i denote the length of a longest directed path from α_1 to α_i (which is well defined since there are no cycles of positive length). By definition of longest paths, $\lambda_i + l_{ij} \leqslant \lambda_j$ for each edge (α_i, α_j). So $\lambda = (\lambda_1, \ldots, \lambda_n)$ is a potential for G. Since all constraints (2.8)b) are linear, the set of all potentials is a convex polyhedron. It is unbounded, since any translation $(\pi_1, \ldots, \pi_n) \pm (d_1, \ldots, d_n)$ of a potential $(\pi_1, \ldots, \pi_n)$ is again a potential.

Finally, the unique minimum non-negative potential is given by the longest path lengths $\lambda_i \geqslant 0$ above, since the iterative application of (2.8)b) along any path from α_1 to α_i yields $\pi_i \geqslant \lambda_i$ for any solution $\pi_1, \ldots, \pi_n$ of (2.8). □

For scheduling applications, the unique componentwise minimum solution of (2.8) has a very natural interpretation. It gives the earliest possible starting times of the activities subject to the contraints (2.8). We call this schedule the *earliest start schedule* associated with the given temporal constraints, and denote it by ES or ES_D, where D is the distance matrix defined below. Moreover, the required vertex α_1 with non-negative path length to all other vertices usually corresponds to an activity representing the start of the project. In the example in fig. 2, we have $ES = (0, 0, 4, 3, 2, 9, 7)$. For instance, ES_7 is determined by the path 1–2–5–7.

The proof of proposition 2.1 relates the test for existence of a time-feasible schedule and the determination of the ES-schedule to the computation of cycle lengths and longest paths in digraphs. This can be done by standard graph algorithms for shortest/longest paths in digraphs, e.g. by the Floyd-Warshall algorithm (cf. [14] for details). It starts with the matrix $D^1 = (d_{ij}^{(1)})$, $i, j = 1, \ldots, n$ of "edge lengths"

$$(2.9) \qquad d_{ij}^{(1)} = \begin{cases} 0 & \text{if } i = j \\ l_{ij} & \text{for each edge } (\alpha_i, \alpha_j) \text{ of } G \\ -\infty & \text{otherwise} \end{cases}$$

and computes the matrix $D = D^{(n+1)}$ according to the following updating for-

mula (in which $u_{ij}^{(m)}$ is the length of a longest path from α_i to α_j that does not use vertices $\alpha_m, \alpha_{m+1}, \ldots, \alpha_n$).

$$(2.10) \qquad d_{ij}^{(m+1)} = \max\left\{ d_{ij}^{(m)},\ d_{im}^{(m)} + d_{mj}^{(m)} \right\}.$$

We then have (cf. for instance [14]):

2.2. PROPOSITION

Let $D^{(0)}$ and D be as defined above and assume that there is a directed path from α_1 to every other α_i. Then:

(1) *Computing D from $D^{(0)}$ according to* (2.10) *takes $O(n^3)$ time.*

(2) *G contains a directed cycle with positive length iff $d_{ii}^{(n+1)} > 0$ for some i.*

(3) *If all $d_{ii}^{(n+1)} = 0$, then $d_{ij}^{(n+1)}$ is the length of a longest path from α_i to α_j in G.*

We call $D = D^{(n+1)}$ the *distance matrix* associated with the temporal constraints (2.8), and denote the $d_{ij}^{(n+1)}$ simply by d_{ij}. Combining the previous results, we obtain:

2.3. COROLLARY

Let D be the distance matrix of a scheduling problem with time windows. Then:

(1) *There exists a time-feasible schedule iff $d_{ii} = 0$ for all $i = 1, \ldots, n$. In that case, $d_{ij} + d_{jk} \leqslant d_{ik}$ for all pairwise distinct jobs $\alpha_i, \alpha_j, \alpha_k$.*

(2) *If all $d_{ii} = 0$, then $S = (S_1, \ldots, S_n) \in \mathbb{R}^n_{\geqslant}$ is a time-feasible schedule iff, for all $i \neq j$, $S_i + d_{ij} \leqslant S_j$.*

(3) *If all $d_{ii} = 0$, then the earliest start schedule ES_D associated with D is given by $ES_D = (d_{11}, d_{12}, \ldots, d_{1n})$.*

Proposition 2.2 and corollary 2.3 state that D can be obtained efficiently, that the diagonal contains information whether a time-feasible schedule exists, and that the first row gives the associated ES_D of all jobs. Moreover, each row i (including $i = 1$) contains the complete information about the minimum temporal distance between the starting time of each job α_j ($j \neq i$) and that of α_i.

In other words, D represents the "transitive closure" of the temporal constraints (2.8). It is *unique* (if all $d_{ii} = 0$, i.e. if $\mathcal{S}_T \neq \emptyset$), while the graph G constructed from the original time lags and windows *is not*. The graph G introduced here is just one way to represent the temporal constraints. Other graph representations (either based on other time lags than start-to-start lags, or using different vertices for S_i and C_i with the constraints $S_i + x_i \leqslant C_i$ and $C_i - x_i \leqslant S_i$) have been used in [2,12,29]. Even with a fixed representation, there may be several different graphs (without redundant "transitive" edges) leading to the same unique distance matrix D and system $\mathcal{S}_T$ of time-feasible schedules. This is why we shall use the *distance matrix D as representative* for the temporal constraints in the standard problem representation introduced below. An additional theoretical justification is given by lemma 3.18.

Table 1
Distance matrix of G in fig. 2

$$D = \begin{pmatrix} 0 & 0 & 4 & 3 & 2 & 9 & 7 \\ -7 & 0 & -3 & 2 & 2 & 4 & 7 \\ -6 & -6 & 0 & -3 & -4 & 5 & 1 \\ -9 & -9 & -5 & 0 & -7 & 2 & 2 \\ -9 & -6 & -5 & 0 & 0 & 2 & 5 \\ -11 & -11 & -7 & -8 & -9 & 0 & -4 \\ -14 & -11 & -10 & -5 & -13 & -3 & 0 \end{pmatrix}$$

For the example of figs. 1 and 2, the associated distance matrix D is given in table 1.

Compared with the order-theoretic case, there are several differences. Modeling usual precedence constraints "$S_i + x_i = C_i \leqslant S_j$" in our model means the introduction of an edge from α_i to α_j with length x_i. Since all $x_i > 0$, there is a time-feasible schedule iff the graph G is acyclic. In that case, the *transitive reduction* of G is the unique non-redundant graph representation, and the *transitive closure* of G is the partial order representing the precedence constraints in the order theoretic model.

If G is acyclic, computation of ES_D is possible in $O(n^2)$ time. However, computation of D (which is the starting point for the solution of the resource constrained problem by branch and bound methods [23,24]) also takes $O(n^3)$ time. Yet another difference can be observed for the convex polyhedron $\mathcal{S}_T$ of time-feasible schedules. If G is acyclic and $S = (S_1, \ldots, S_n)$, $S' = (S'_1, \ldots, S'_n) \in S_T$, then also $S + S' = (S_1 + S'_1, \ldots, S_n + S'_n) \in S_T$ and, for $\lambda > 1$, $\lambda \cdot S = (\lambda S_1, \ldots, \lambda S_n) \in S_T$. These properties of being closed under vector addition and (restricted) scalar multiplication is lost as soon as G contains directed cycles.

Finally it should be noted that the approach taken here remains also valid for more general temporal constraints. The essential requirement is only that they can be represented as $S_i + l_i j \leqslant S_j$. The l_{ij} may, however, be arbitrary functions of the processing times $x_1, \ldots, x_n$. In this respect, the transformation rules (2.4)–(2.7) just represent a special case, in which each l_{ij} is an affine-linear function of x and x_j. Of course, with respect to sensitivity analysis or stability behaviour (cf. sect. 3.5) certain smoothness properties of the l_{ij} as a function of the x_i will be required.

2.3. RESOURCE CONSTRAINTS

While being processed, the jobs require certain resources from a set I of resource types. Resources are *reusable*, i.e. they are released when they are no longer required by a job and are then available for the processing of any other job.

In its most general form, the demand of job $\alpha_j \in A$ for resource $i \in I$ is given by a *demand profile*, i.e. a piecewise-constant function r_{i,α_j}: $[0, x_j] \to \mathbb{N} = \{0, 1, 2, \ldots\}$, where $r_{i,\alpha_j}(t)$ denotes the number of units of resource i required by job α_j at time t after its starting time S_j.

Each resource type $i \in I$ is only available in a limited amount. Similar to the requirements, the availability of resource $i \in I$ is generally given by a *supply profile*, i.e. a piecewise-constant function R_i: $[0, \infty] \to \mathbb{N}$, where $r_i(t)$ denotes the number of units available of resource i at time t of the project execution.

A schedule S is called *resource-feasible* with respect to the r_{i,α_j} and R_i ($i \in I$, $\alpha_j \in A$) if at each time t of the project execution, the demand of each resource $i \in I$ does not exceed the momentary supply $R_i(t)$, i.e.

$$\sum_{S_j \leqslant t < C_j} r_{i\alpha_j}(t - S_j) \leqslant R_i(t) \text{ for each } t \text{ and every } i \in I. \tag{2.11}$$

By $\mathcal{S}_R$ we denote the set of all resource-feasible schedules.

To avoid trivial non-feasible cases, it is convenient to assume that each individual job α_j can (not regarding the temporal constraints) be scheduled without interruption in some interval, i.e. there is some $t_o \geqslant 0$ such that

$$r_{i\alpha_j}(t) \leqslant R_i(t_o + t) \quad \text{for all } t \in [0, x_j] \text{ and each } i \in I. \tag{2.12}$$

Since this condition can be verified efficiently, it has no influence on the complexity of the whole problem (which is NP-hard, cf. sect. 3).

A schedule S is called *feasible* if it is both time-feasible and resource-feasible. Then $\mathcal{S} = \mathcal{S}_T \cap \mathcal{S}_R$ denotes the set of all feasible schedules.

A very important property of our model is the fact that the *variable* resource demands and supplies can already be modeled by *constant* demand and supply profiles. The corresponding reductions are described in the sequel. They make appropriate use of time windows and dummy jobs without increasing the input length of the problem description.

Let $r_{i\alpha_j}$ be a demand profile. Since demand profiles are piecewise constant, $d_{i\alpha_j}$ has constant values $c_1, \ldots, c_m$ on subintervals $[0, y_1[, [y_1, y_2[, \ldots, [y_{m-1}, x_j]$ of $[0, x_j]$, respectively (where $c_k \neq c_{k+1}$; $k = 1, \ldots, m-1$). Split the job α_j into m parts $\alpha_j^1, \ldots, \alpha_j^m$ with processing times $y_1, y_2 - y_1, \ldots, x_j - y_{m-1}$ and constant resource demands $c_1, \ldots, c_m$, respectively. Respect the non-preemption condition by introducing the temporal constraints $C(\alpha_j^k) = S(\alpha_j^{k+1})$, $k = 1, \ldots, m-1$. So in terms of start-to-start lags, this means that

$$\begin{cases} S(\alpha_j^k) + y_k - y_{k-1} \leqslant S(\alpha_j^{k+1}) \text{ and} \\ S(\alpha_j^{k+1}) - y_k + y_{k-1} \leqslant S(\alpha_j^k). \end{cases} \tag{2.13}$$

Obviously, this process of splitting jobs and gluing the parts together by temporal constraints makes all resource demands constant for one given demand profile. An example is given in fig. 3.

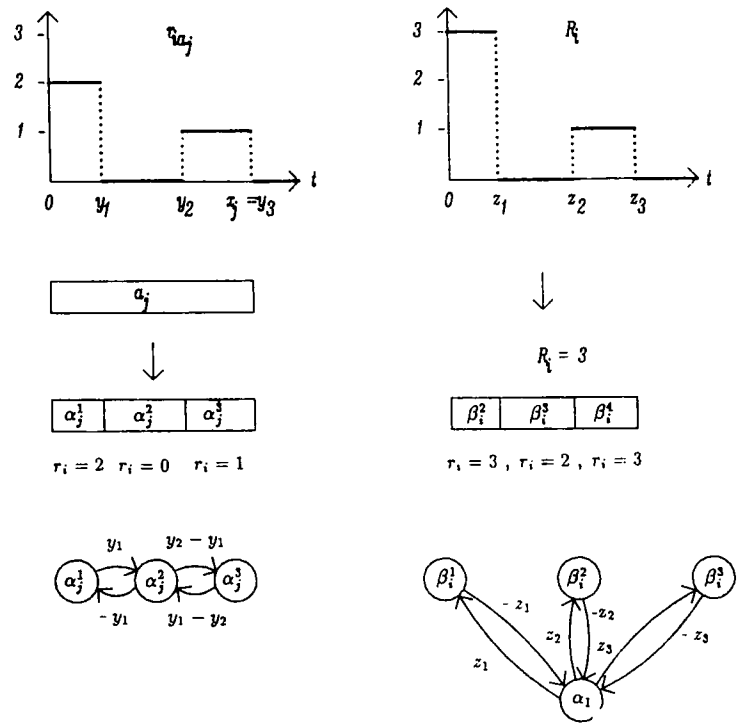

Fig. 3. Transformation of variable demand and supply profiles.

If job α_j has several non-constant demand profiles, then take the largest intervals on which each profile $r_{i\alpha_j}$ is constant as the intervals $[0, y_1[, [y_1, y_2[, \ldots$ above and proceed analogously. Then ob α_j will be split into at most as many subjobs as there are different values c_k in the union of the ranges of all $r_{i\alpha_j}$, $i \in I$.

For a supply profile R_i, we proceed as follows (see also fig. 3). Let $[0, z_1], [z_1, z_2], \ldots, [z_m, \infty]$ be the maximal subintervals on which R_i is constant, and let $b_1, \ldots, b_{m+1}$ be the associated values. The idea then is to introduce for each subinterval $[z_{k-1}, z_k]$ with $b_i < \max\{b_1, \ldots, b_{m+1}\}$ a dummy job β_k with processing time $x(\beta_k) = z_k - z_{k-1}$, constant resource demand $r_{i\beta_k} = \max\{b_1, \ldots, b_{m+1}\} - b_i$, and temporal constraints $S(\beta_k) = S(\alpha_1) + z_{k-1}$ that fix its processing period exactly in the interval $[z_k - 1, z_k]$. So loosely speaking, we introduce a dummy job for each supply interval with "low" supply that must be processed during this interval at project execution and absorbs the nonavailable resource units. This allows us to set $R_i = \max\{b_1, \ldots, b_{m+1}\}$ during project execution. Obviously, we need at most as many dummy jobs for R_i as there are different values b_k in the range of R_i.

Let P be the original problem on the set $A = \{\alpha_1, \ldots, \alpha_n\}$ with the variable resource profiles, and let $\bar{P}$ denote the transformed problem on the set $A^* =$

$\{\alpha_1, \ldots, \alpha_n, \alpha_{n+1}, \ldots, \alpha_m\}$, where $\alpha_{n+1}, \ldots, \alpha_m$ denote the newly introduced jobs (resulting from either job splitting or adding dummies).

Let $\mathcal{S}$ and $\mathcal{S}^*$ denote the sets of feasible schedules for P and P^* respectively. We then have:

2.4. THEOREM

$\mathcal{S}$ is the projection of $\mathcal{S}^*$ under the projection $pr_{\{1,\ldots,n\}}: \mathbb{R}^m \to \mathbb{R}^n$ with $pr_{\{1,\ldots,n\}}(y_1, \ldots, y_m) = (y_1, \ldots, y_n)$; i.e. if $S^* = (S_1, \ldots, S_n, S_{n+1}, \ldots, S_m) \in \mathcal{S}^*$, then $S = (S_1, \ldots, S_n) \in \mathcal{S}$, and each schedule $S \in \mathcal{S}$ is obtained in that way.

Proof

Straightforward from the transformation rules. □

Note that this reduction is only possible by changing both the *resource* and the *temporal* constraints. So with respect to the representation $\mathcal{S} = \mathcal{S}_T \cap \mathcal{S}_R$, the constraints defining $\mathcal{S}_R$ are relaxed, while those defining $\mathcal{S}_T$ are strengthened.

Concerning the complexity to carry out the transformation and its influence on the length of the encodings of P and P^* observe that, for P, we need to store all the subintervals of the $r_{i\alpha_j}$ and the R_i and the associated values c_k and b_l. Since we introduce at most that many additional jobs and twice that many additional temporal constraints, we obtain:

2.5. THEOREM

The number of additional jobs in P^* is bounded by the total number N of values of the demand and supply profiles of P (and thus by the input length of P).

The transformation requires at most $O(N)$ time and changes the input length only by a constant factor.

Proof

Straight-forward from the above remarks and the transformation rules. □

So altogether, we have:

2.6. COROLLARY

Each scheduling problem with time windows and variable resource demand and supply is polynomially equivalent to a scheduling problem with time windows and constant resource demand and supply.

As in the order-theoretic case, constant resource demand and supply profiles permit a simple–and theoretically very convenient–description of the resource constraints by means of a system of forbidden sets.

Let $r_{i\alpha j} \in \mathbb{N}$ and $R_i \in \mathbb{N}$ denote the constant resource demand and supply of sort $i \in I$. Then a set $N \subseteq A$ of jobs is called *forbidden* if

$$\sum_{\alpha_j \in N} r_{i,\alpha_j} > R_i \text{ for some } i \in I, \tag{2.14}$$

i.e. if, for some $i \in I$, the total amount of resource i required by the jobs from N exceeds the available amount R_i. Note that (2.14) is just the negation of (2.11), which is independent of time t because of the constant demands and supplies.

The system of all forbidden sets is denoted by $\mathcal{N}$. In terms of the system $\mathcal{N}$, a schedule $\mathcal{S}$ is resource-feasible iff no set $N \in \mathcal{N}$ is scheduled simultaneously at any time t, i.e.

$$(2.15) \quad N \not\subseteq S(t) := \{\alpha_j \in A \mid S_j \leqslant t < C_j\} \quad \text{for all } N \in \mathcal{N}.$$

It follows easily that forbidden sets characterize exactly all kinds of time-independent resource constraints (see e.g. [23]). We will see later that they model the "essential conflicts" in simultaneity that have to be settled by any feasible schedule.

The only disadvantage of the description by forbidden sets is the fact that the number of sets required may be exponential in the input length, even if we restrict ourselves to the minimal forbidden sets (w.r.t. set inclusion.) However, practical experience [2] shows that their determination usually causes no problem, in particular when compared with the time required to find an optimal solution. Moreover, in many applications the availabilities R_i are constraint and independent of n, thus resulting in a size of $\mathcal{N}$ that is polynomial in n.

Note, however, that even for polynomial $\mathcal{N}$, the size of $\mathcal{N}$ and the size of the sets in $\mathcal{N}$ significantly influence the running time of the branch and bound algorithm in Section 4. So altogether, since we will use $\mathcal{N}$ essentially in our algorithmic methods, we prefer to neglect the disadvantage of a possibly "long" problem description and use the system of forbidden sets for representing the resource constraints in our standard model.

2.4. THE STANDARD REPRESENTATION

As a consequence of the preceding considerations, each scheduling problem considered in this paper can be represented by

- a set $A = \{\alpha_1, \ldots, \alpha_n\}$ of jobs
- a vector $x = (x_1, \ldots, x_n)$ of processing times
- a distance matrix D representing the temporal constraints (2.8)
- a system $\mathcal{N}$ of forbidden sets representing the resource constraints
- a regular cost function κ

We refer to this as the *standard representation* of the scheduling problem, and denote it by $[A, x, D, \mathcal{N}, \kappa]$.

The optimization aim then consists in computing the least cost (the *optimum value*)

$$(2.16) \quad \rho(\kappa; x) := \inf\{\kappa(S; x) \mid S \in \mathcal{S}\},$$

and in determining an *optimal schedule* S, i.e. a schedule S with $\rho(\kappa; x) = \kappa(S; x)$.

Here $\mathcal{S}=\mathcal{S}_T \cap \mathcal{S}_R$ denotes the set of all feasible schedules are defined in sects. 2.2 and 2.3.

The reduction of a variety of scheduling problems to this standard representation is not just for reasons of unification. We will see that this representation does, indeed, reveal the basic underlying mathematical structure of scheduling problems, and serves as a "natural" starting point for both theoretical insights and algorithmic procedures.

This is the reason why the standard representation is also a natural *internal model* for computer packages or DSS for dealing with real-life scheduling problems. The internal model may be completely hidden from the user (for whom it may in mathematical respect be much too complicated to understand). Interaction with the user would then take place via a user interface that transforms the user parameters (such as time lags, precedence constraints, resource demands and supplies etc.) into the internal model according to the transformation rules given in this section, and retransforms results, messages etc. into the user's representation. All internal computations, however, would be based on the "suitable" internal model.

A possible specification of such a package is given in [4] as part of an advanced decision support system for the interactive solution of scheduling problems.

3. The structural approach

3.1. THE EMBEDDED PARTIAL ORDER

Let P be a scheduling problem in standard form $[A, x, D, \mathcal{N}, \kappa]$. Assume that there exists a time-feasible schedule, i.e. $\mathcal{S}_T \neq \emptyset$. We will show that the conditions on $\mathcal{S}_T$ represented by D contain certain "embedded" partial order constraints. This partial order represents precedence constrains that any time-feasible schedule must obey, and in addition represents with its antichains all sets of jobs that may be scheduled simultaneously. It will serve as the central structure for the solution methods developed in this paper and relates the general case to the order-theoretic approach.

Define the relation $<_D$ by

(3.1) $\alpha_i <_D \alpha_j$ if every time-feasible schedule S fulfils $C_i \leqslant S_j$.

So $\alpha_i <_D \alpha_j$ iff α_j can only be started after α_i has been completed. It is clear that $<_D$ is a partial ordering on the set of jobs. We call $\Theta_D = (A, <_D)$ the *embedded partial order* of the temporal constraints. The following result shows that Θ_D is, indeed, "embedded" into the distance matrix D.

3.1. THEOREM

$\alpha_i <_D \alpha_j$ *iff* $x_i \leqslant d_{ij}$.

This shows that D contains all the information about Θ_D and that Θ_D can be obtained from D in $O(n^2)$ time. The proof of this result is based on two lemmas about schedule modification.

3.2. LEMMA
Let S be a time-feasible schedule. Then for any $\alpha_i \in A$ and any $d \in \mathbb{R}^1$, S' defined by

$$(3.2) \qquad \begin{cases} S' = S_i + d \\ S'_j = \max\{S_j,\ S_i + d_{ij} + d\} \quad \text{if } j \neq i \end{cases}$$

is also a time feasible schedule.

Proof
Assume that S' is not time-feasible. Then there are $\alpha_l, \alpha_m \in A$ with $S'_l + d_{lm} > S'_m$. Since $S_l \leqslant S'_l$, this is only possible if $S'_l = S_i + d_{il} + d$. So

$$S_i + d_{il} + d + d_{lm} > S'_m \geqslant S_i + d_{im} + d,$$

which implies $d_{il} + d_{lm} > d_{im}$, a contradiction to corollary 2.3 (1). □

3.3. LEMMA
Let S be a time-feasible schedule. Let $U = \{\alpha_1, \ldots, \alpha_k\}$ be a set of jobs with $S_i \leqslant S_k$ and $x_i > d_{ij}$ for all $i,\ j = 1, \ldots, k$ $(i \neq j)$.
Then there is a time-feasible schedule S' with $S_k = S'_k$ and $S'_i \leqslant S'_k < C'_i$ for $i = 1, \ldots, k-1$.

Proof
Let (in the order $\alpha_1, \alpha_2, \ldots, \alpha_k$) α_i be the last job with $C_i \leqslant S_k$. (If there is no such job, then already S has the required property.)
We will modify S into a time-feasible schedule S' with $S_j \leqslant S'_j \leqslant S'_k = S_k$ $(j = 1, \ldots, k-1)$ and $C'_i > S'_k$. Clearly, a finite sequence of such modifications will construct a time-feasible schedule with the required properties.
To this end, put $d := S_k - C_i + \delta$, where

$$\delta := \min\{x_i,\ \min\{x_i - d_{ij} \mid j = 1, \ldots, k,\ j \neq i\}\}.$$

Note that $\delta > 0$. By lemma 3.2, S' defined by

$$S'_i := S_i + d$$

$$S'_j := \max\{S_j,\ S_i + d_{ij} + d\} \text{ if } j \neq i$$

is a time-feasible schedule with $S_j \leqslant S'_j$ for all α_j $(j = i, \ldots, k)$. We will show that S' has the claimed properties.
First, for $\alpha_j \in \{\alpha_1, \ldots, \alpha_k\} \setminus \{\alpha_i\}$ we obtain

$$S_i + d_{ij} + d = S_i + d_{ij} + S_k - C_i + \delta = S_k - (x_i - d_{ij}) + \delta \leqslant S_k$$

by definition of δ. The definition of S' then implies $S_k = S'_k$ and $S'_j \leqslant S_k = S'_k$ for $j \in \{1, \ldots, k\} \setminus \{i\}$.

For α_i, we have

$$S'_i = S_i + d = S_i + S_k - C_i + \delta = S_k - x_i + \delta \leqslant S_k$$

by definition of δ.

Finally, concerning the completion time C'_i of α_i, we have

$$C'_i = C_i + d = S_k + \delta > S_k,$$

again by definition of δ. □

Proof of theorem 3.1

Let $x_i \leqslant d_{ij}$. Then, for any time-feasible schedule S,

$$C_i = S_i + x_i \leqslant S_i + d_{ij} \leqslant S_j$$

because of corollary 2.3 (1). Hence $\alpha_i <_D \alpha_j$.

In the converse direction, let $\alpha_i <_D \alpha_j$, and let S be a time-feasible schedule. By definition of $<_D$, $C_i \leqslant S_j$, and thus $S_i \leqslant S_j$ and $d_{ji} \leqslant 0$, which implies $x_j \geqslant d_{ji}$. Assume that $x_i > d_{ij}$. Then lemma 3.3 can be applied to $U = \{\alpha_i, \alpha_j\}$, i.e. there is a time-feasible schedule S' with $S'_i \leqslant S'_j < C'_i$, a contradiction to $\alpha_i <_D \alpha_j$. Hence $x_i \leqslant d_{ij}$. □

The proof shows that missing precedence constraints $\alpha_i <_D \alpha_j$ related to simultaneous scheduling. In fact, there is a very strong such relationship, which is formulated in the next theorem.

Call a set U of jobs *potentially parallel* or simply *parallel* if there exists a time-feasible schedule S and a time t such that $S_i \leqslant t < C_i$ for all $\alpha_i \in U$ (i.e. all jobs from U are scheduled *in parallel* or *simultaneously* at time t with respect to S). Call a set U of jobs an *antichain* of a partial order $\Theta = (A, <_\Theta)$ on A, if any two jobs $\alpha_i, \alpha_j \in U$ are *incomparable* or *unrelated* (i.e. neither $\alpha_i <_\Theta \alpha_j$ nor $\alpha_j <_\Theta \alpha_i$). We denote this by $\alpha_i \|_\Theta \alpha_j$.

3.4. THEOREM

Let $P = [A, x, D, \mathcal{N}, \kappa]$ be given with $\mathcal{S}_T \neq \emptyset$. Then the parallel sets of jobs of P are exactly the antichains of the embedded partial order $\Theta_D = (A, <_D)$.

Proof

Let U be parallel, and let S be the associated schedule that achieves the simultaneous scheduling of U. Then $C_i > S_j$ and $C_j > S_i$ for any two jobs $\alpha_i, \alpha_j \in U$, and thus $\alpha_i \|_D \alpha_j$ by (3.1). It follows that U is an antichain of Θ_D.

In the converse direction, let $U = \{\alpha_1, \ldots, \alpha_k\}$ be an antichain of Θ_D. Let S be a time-feasible schedule for P and assume w.l.o.g. that $S_j \leqslant S_k$, $j = 1, \ldots, k-1$. Since U is an antichain, $\alpha_j \|_D \alpha_k$ $(j = 1, \ldots, k-1)$ and thus $x_j > d_{jk}$ for each $j = 1, \ldots, k-1$ because of theorem 3.1. So lemma 3.3 applies to U and yields a schedule S' that schedules U simultaneously. □

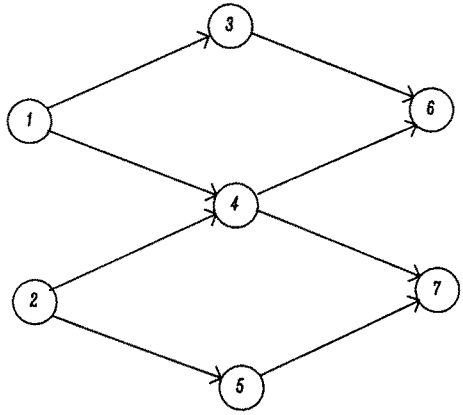

Fig. 4. Node diagram of the embedded partial order of Θ from example 3.5.

As a consequence of theorem 3.4, it suffices with respect to the resource constraints, to specify only those (minimal) forbidden sets that are antichains of the embedded partical order Θ_D. This follows immediately from the fact that only antichains of Θ_D are parallel sets. We will denote the system of $\subseteq$-minimal forbidden sets that are antichains of Θ_D by $\mathcal{N}_{\text{red}}$ and refer to it as the system of *reduced forbidden sets*.

Note that $\mathcal{N}_{\text{red}}$ can be determined completely analogously to the order-theoretic case from the (constant) demand and supply profiles by traversing Θ_D in a special way that generates all maximal antichains of Θ_D. See Bartusch [2] for details of this method.

Finally, it should be noted that in the order-theoretic case, the embedded partial order Θ_D is just the given partial order of technological precedence constraints.

We close this subsection with an example.

3.5. EXAMPLE

Consider the problem of fig. 1 whose distance matrix D is given in table 1. Comparing D with $x = (3, 1, 4, 2, 5, 2, 1)$, one obtains $x_i \leqslant d_{ij}$ and thus $\alpha_i <_D \alpha_j$ for $(i, j) \in \{(1, 3), (1, 4), (1, 6), (1, 7), (2, 4), (2, 5), (2, 6), (2, 7), (3, 6), (4, 6), (4, 7), (5, 7)\}$. The embedded partial order Θ_D is given in fig. 4. The maximal antichains of Θ_D are $\{1, 2\}$, $\{1, 5\}$, $\{2, 3\}$, $\{3, 4, 5\}$, $\{3, 7\}$, $\{5, 6\}$, and $\{6, 7\}$. In the earliest-start schedule ES, the antichains $\{1, 2\}$, $\{1, 5\}$, $\{3, 4, 5\}$ and $\{3, 7\}$ are scheduled in parallel (see fig. 5). The remaining antichains are scheduled in parallel by the schedules $S^{(1)}$ and $S^{(2)}$ of fig. 5.

$S^{(1)}$ is obtained from ES by applying the construction of lemma 3.3 tc $U = \{2, 3\}$. This yields $\delta = \min\{x_2, x_2 - d_{23}\} = \min\{1, 1 + 4\} = 1$ and $d = S_3 - C_2 + \delta = 4 - 1 + 1 = 4$. So $\{2, 3\}$ is scheduled in parallel, and, as a side effect

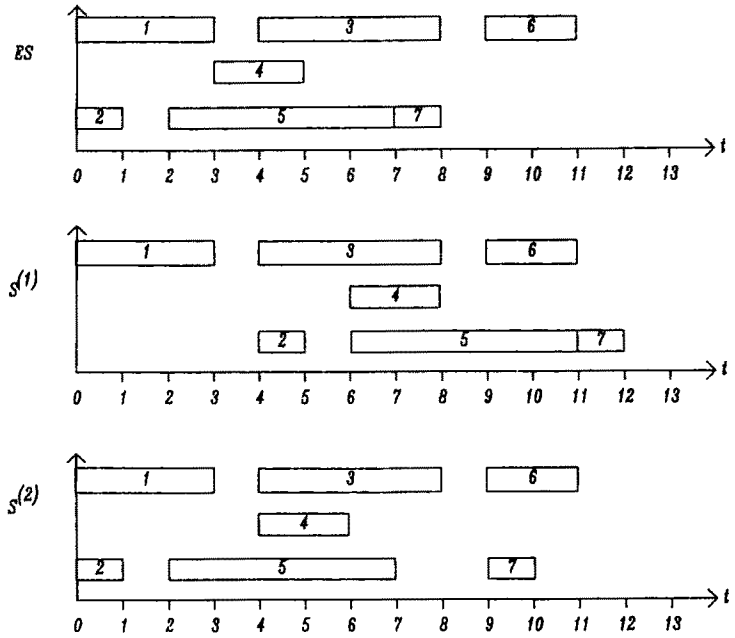

Fig. 5. Some time-feasible schedules for example 3.5.

also $\{5, 6\}$. $S^{(2)}$ is then again obtained from ES by applying the construction to $U = \{7, 6\}$. Note that in both cases, the start of job 4 is postponed because of $d_{76} = -5$.

3.2. THE MAIN REPRESENTATION THEOREM

The embedded partial order Θ_D derived in the previous subsection expresses only properties of *time-feasibility*. We will now investigate how *resource-feasibility* is related to Θ_D. Our analysis yields some additional, strong relationships with the order-theoretic case.

It is well known that any schedule S for a scheduling problem on A induces a partial order $\Theta_S = (A, <_S)$ on A by putting

$$(3.3) \qquad \alpha_i <_S \alpha_j \quad \text{iff } C_i \leqslant S_j,$$

i.e. if α_i is completed before α_j is started. Θ_S is an *interval order* and reflects the precedences between jobs created by the schedule S. (See [19] or [25] for more information about interval orders and their relationship with scheduling problems.) We call Θ_S the partial order *induced* by the schedule S.

In particular, any *feasible* schedule S induces a partial order Θ_S. We will now investigate how these partial orders are related to the embedded partial order Θ_D.

3.6. LEMMA

Let $P = [A, x, D, \mathcal{N}, \kappa]$ *be a scheduling problem, and let* S *be a feasible schedule for* P. *Then*

(1) $\alpha_i <_D \alpha_j \Rightarrow \alpha_i <_S \alpha_j$

(2) *For each forbidden set* $N \in \mathcal{N}$, *there are jobs* $\alpha_i, \alpha_j \in N$ *with* $\alpha_i <_S \alpha_j$.

(3) *The temporal constraints given by*

$$(3.4) \qquad \begin{cases} t_i + d_{ij} \leqslant t_j & \text{for all } \alpha_i \neq \alpha_j \\ t_i + x_i \leqslant t_j & \text{for all } \alpha_i <_S \alpha_j \end{cases}$$

are solvable.

(4) *The earliest-start schedule* $ES_{D+\Theta_S}$ *associated with the temporal constraints* (3.4) *is a feasible schedule for* P *with* $ES_{D+\Theta_S} \leqslant S$ (*componentwise*) *and* $\kappa(ES_{D+\Theta_S};\ x) \leqslant \kappa(S;\ x)$.

Proof

Any feasible schedule must respect the constraints (3.1) represented by the embedded partial order. So $\alpha_i <_\Theta \alpha_j$ yields $C_i \leqslant S_j$ and thus $\alpha_i <_{\Theta_S} \alpha_j$. This proves (1).

Since S is feasible, no forbidden set $N \in \mathcal{N}$ can be parallel with respect to S. Hence for any $N \in \mathcal{N}$, there are $\alpha_i, \alpha_j \in N$ with $C_i \leqslant S_j$, and thus $\alpha_i <_{\Theta_S} \alpha_j$ by definition of Θ_S.

S itself is a solution to (3.4). So by proposition 2.1 and corollary 2.3, $ES_{D+\Theta_S}$ is well defined and componentwise better than S. $ES_{D+\Theta_S}$ is feasible for P since it preserves the temporal constraints given by D (first line of (3.4)) and does not schedule a forbidden set in parallel. This last property follows from the second line of (3.4) together with part (2) of the lemma.

Finally, the monotonicity of κ then yields that $ES_{D+\Theta_S}$ is less costly than S. □

Lemma 3.6 shows that each feasible schedule can be improved upon by an *ES*-schedule arising from a certain "extension" of Θ_D whose properties are given by (2) and (3). This motivates the following definitions:

Let $\Theta_1 = (A, <_1)$ and $\Theta_2 = (A, <_2)$ be partial orders on the same set A. Θ_2 is called an *extension* of Θ_1 (denoted by $\Theta_1 \subseteq \Theta_2$) if, for any $\alpha, \beta \in A$, $\alpha <_1 \beta$ implies that $\alpha <_2 \beta$. An extension $\Theta = (A, <_\Theta)$ of the embedded partial order Θ_D of some scheduling problem $P = [A, x, D, \mathcal{N}, \kappa]$ is called *resource-feasible* if

(3.5) no antichain of Θ is a forbidden set,

and *time-feasible* if the system $D + \Theta$ of temporal constraints given by

$$(3.6) \qquad \begin{cases} S_i + d_{ij} \leqslant S_j & \text{for all } d_{ij},\ i \neq j \\ S_i + x_i \leqslant S_j & \text{for all } \alpha_i <_\Theta \alpha_j \end{cases}$$

has a solution.

If Θ is time-feasible, then the earliest start schedule associated with the system (3.6) is denoted by $ES_{D+\Theta}$.

Finally, Θ is called *feasible*, if it is both time- and resource-feasible.

With these notations, statements (1)–(3) of lemma 3.6 may be restated by saying that induced partial orders Θ_S of feasible schedules S are feasible. The converse direction is considered in the next lemma.

3.7. LEMMA

Let Θ be a feasible extension of the embedded partial order Θ_D of $P = [A, x, D, \mathcal{N}, \kappa]$. Then $ES_{D+\Theta}$ is a feasible schedule for P.

Proof

Let Θ be feasible. Then $ES_{D+\Theta}$ respects the temporal constraints given by D because of (3.6) and hence is *time-feasible* for P. Property (3.5) implies that to each $N \in \mathcal{N}$ there are two comparable jobs, say $\alpha_i, \alpha_j \in N$ with $\alpha_i <_\Theta \alpha_j$. Then (3.6) implies that α_i is completed before α_j is started with respect to $ES_{D+\Theta}$. Hence N is not scheduled simultaneously and thus $ES_{D+\Theta}$ is also *resource-feasible* for P. □

Altogether, we obtain:

3.8. THEOREM: (MAIN REPRESENTATION THEOREM)

Let $P = [A, x, D, \mathcal{N}, \kappa]$ be a scheduling problem. Then the optimum value $\rho(\kappa; x)$ has the representation

$$\rho(\kappa, x) = \min\{\kappa(ES_{D+\Theta}; x) \mid \Theta_D \subseteq \Theta, \Theta \text{ feasible}\}. \tag{3.7}$$

Proof

Denote the left-hand-side and the right-hand-side of (3.7) by ρ and ρ', respectively. Then lemma 3.7 and (2.16) imply that $\rho \leqslant \rho'$. Conversely, lemma 3.6 shows that any feasible schedule S can be improved upon by taking $ES_{D+\Theta_S}$. Hence $\rho \geqslant \rho'$. □

Theorem 3.8 provides many theoretical insights into the optimization problem given by $P = [A, x, D, \mathcal{N}, \kappa]$.

First of all, it shows that only finitely many schedules need to be considered. They arise from feasible extensions of the embedded partial order Θ_D, which – as will be shown in the next subsection – constitute a subclass of the semilattice of all partial orders on A with interesting properties that determine e.g. the geometrical nature of the set of all feasible schedules and the analytical properties of the optimum value $\rho(\kappa, \cdot)$ as a function of the processing times x.

Second, theorem 3.8 is the basis for the algorithmic methods developed in sect. 4, which essentially consists in constructing in a branch-and-bound approach a

feasible extension Θ of Θ_D for which $ES_{D+\Theta}$ is optimal for P. The necessary theoretical consequences of theorem 3.8 for this approach are derived in subsection 3.4.

The representation theorem 3.8 and its consequences can be viewed as a natural generalization of the order-theoretic approach for precedence constrained scheduling problems [18,20,23,24]. In fact, this approach was the motivation for the treatment given here, and the notions of *embedded partial order* and *feasible extension* presented here turn out to provide the right concept for a far-reaching analogy.

There are, however, some important differences that also cause certain nasty effects with respect to the feasibility domain and the behaviour of the function $\rho(\kappa; \cdot)$. Mainly, these differences arise from the necessity to require time-feasibility of "feasible" extensions in addition to resource-feasibility, which is not necessary for precedence-constrained scheduling. As our treatment below will show, this requirement causes some nasty time-dependencies of the feasibility domain. Compared with precedence-constrained scheduling, this results in weaker properties of $\rho(\kappa; \cdot)$ and the feasibility domain, but, on the other hand, leads to stronger algorithmic properties.

3.3. THE SET OF FEASIBLE EXTENSION

Given $P = [A, x, D, \mathcal{N}, \kappa]$, let $\mathcal{F}_T$, $\mathcal{F}_R$ and $\mathcal{F}$ denote the set of time-feasible, resource-feasible, and feasible extensions of Θ_D, respectively. Furthermore, let $\mathcal{O}(A)$ denote the set of all partial orders on A. It is well known that $\mathcal{O}(A)$ ordered by " $\subseteq$ " is a semi-lattice with certain structural properties, see e.g. [23].

We will now investigate how $\mathcal{F}$ is embedded into $\mathcal{O}(A)$.

3.9. THEOREM

For $\mathcal{F}_T$, $\mathcal{F}_R$, and $\mathcal{F}$, the following properties hold:

(1) *$\mathcal{F}_T$ is a convex subset of $\mathcal{O}(A)$ with Θ_D as least element,*

(2) *$\mathcal{F}$ is a filter of $\mathcal{O}(A)$,*

(3) *$\mathcal{F} = \mathcal{F}_T \cap \mathcal{F}_R$ is a convex subset of $\mathcal{O}(A)$.*

Proof

To show (1), let $\Theta \in \mathcal{F}_T$. Then the system $D + \Theta$ given by (3.6) has a solution. Obviously, any such solution also satisfies the system $D + \Theta'$ for any Θ' with $\Theta_D \subseteq \Theta' \subseteq \Theta$. This proves (1).

Statement (2) follows immediately from (3.5), since for any extension Θ' of some $\Theta \in \mathcal{F}_R$, each antichain U of Θ' is also an antichain of Θ and thus $U \notin \mathcal{N}$.

Since $\mathcal{F} = \mathcal{F}_T \cap \mathcal{F}_R$, and any filter is convex, $\mathcal{F}$ is convex, too. □

Note that, different from precedence-constrained scheduling, $\mathcal{F}$ is in general not a filter, i.e. there may be resource-feasible extensions Θ of Θ_D that are not

time-feasible. We will see in the next sections that this behaviour is closely related to the existence of "cycles" in the constraints (3.6) represented by $D + \Theta$.

As a direct consequence, one obtains that testing for the existence of a feasible extension – or equivalently, testing whether the nonempty sets $\mathscr{F}_T$ and $\mathscr{F}_R$ intersect – is already NP-hard, in general. Concerning computational complexity, this establishes a severe difference with precedence-constrained scheduling.

3.10. THEOREM

Testing whether a scheduling problem $P = [A, x, D, \mathcal{N}]$ *has a feasible solution is an NP-hard problem, even in the case when* $x = (1, 1, \ldots, 1)$ *and* $\mathcal{N}$ *represents machine-constraints (i.e.* $\mathcal{N} = \{N \subseteq A: |N| \geq m + 1$ *for some fixed* $m < n$.)

Proof

In order to avoid problems with a possible "exponential" representation of $\mathcal{N}$, we will represent the machine constraints just by the number m of available machines. Furthermore, D will contain only values $d_{ij} \in \{0, 1, \ldots, n\} \cup \{-2\}$ and the symbol "$-\infty$". So the input-length for such a problem $P = [A, D, m]$ is $O(n^2 \log n)$.

We will use the fact that the following precedence-constrained unit-time scheduling problem Q is NP-complete; see [16] or [11] for details:

Q: INSTANCE: A partial order Θ_o on $A = \{\alpha_1, \ldots, \alpha_n\}$, processing times $x_j = 1$ $(j = 1, \ldots, n)$, a number m of parallel machines.

QUESTION: Does there exist a feasible schedule on m machines with length $l \leq 3$?

In [16], there is a direct reduction of CLIQUE to Q. We show now that $Q \propto P$. Define $D^{(o)}$ by

$$d_{ij}^{(o)} = \begin{cases} 1 & \text{if } \alpha_i <_{\Theta_o} \alpha_j \\ -2 & \text{if } \alpha_j <_{\Theta_o} \alpha_i \\ 0 & \text{if } i = j \\ -\infty & \text{otherwise,} \end{cases} \tag{3.8}$$

and let D be the associated distance matrix in the sense of proposition 2.2. The constraints (3.8) are just the precedence constraints $\alpha <_{\Theta_o} \alpha_j$ and the constraint that there may be at most a time lag of 2 between α_i, α_j with $\alpha_i <_{\Theta_o} \alpha_j$ (which of course implies that any time-feasible schedule respects Θ_o and has length $l \leq 3$).

Obviously, there is a feasible schedule with length $l \leq 3$ iff there is a feasible extension Θ of $\Theta_o = \Theta_D$, i.e. an extension with $|U| \leq m$ for all antichains U of Θ. □

The reason for this NP-hardness result is that "deadlines" can be modeled by maximum time-lags, thus transforming feasibility problems for precedence-constrained scheduling with deadlines to scheduling problems with time windows.

We close this subsection with a monotonicity property on $\mathcal{O}(A)$ that is essential for the algorithmic approach developed in section 4.

Let $D_1 = (d_{ij}^{(1)})$ and $D_2 = (d_{ij}^{(2)})$ be two $n \times n$ (distance) matrices. We write $D_1 \leqslant D_2$ iff $d_{ij}^{(1)} \leqslant d_{ij}^{(2)}$ for all pairs i, j. Similarly, for two solvable constraint system $D + \Theta_1$, $D + \Theta_2$, where Θ_1, Θ_2 are time-feasible extensions of Θ_D, we write $D + \Theta_1 \leqslant D + \Theta_2$ if, for the distance matrices D_i representing $D + \Theta_i$ $(i = 1, 2)$ in the sense of proposition 2.2, we have $D_1 \leqslant D_2$. Finally, we call a distance matrix D' *feasible*, if it is the distance matrix of a constraint system $D + \Theta$ with feasible extension Θ of Θ_D.

3.11. LEMMA

Let Θ_1, Θ_2 be time-feasible extensions of Θ_D. Then

$$\Theta_1 \subseteq \Theta_2 \Rightarrow D + \Theta_1 \leqslant D + \Theta_2 \tag{3.10}$$

In particular, $ES_{D+\Theta_1} \leqslant ES_{D+\Theta_2}$ and $\kappa(ES_{D+\Theta_1}; x) \leqslant \kappa(ES_{D+\Theta_2}; x)$ for any cost function κ.

Proof

Let $\Theta_1 \subseteq \Theta_2$ and let $D_s = (d_{ij}^{(s)})$, $s = 1, 2$, denote the distance matrices of the (solvable) systems $D + \Theta_s$. Since $\Theta_1 \subseteq \Theta_2$, any time-feasible schedule S for $D + \Theta_2$ is also time-feasible for $D + \Theta_1$. Suppose that (3.10) is false, i.e. there are $k \neq l$ with $d_{kl}^{(1)} > d_{kl}^{(2)}$. Let S be a time-feasible schedule for D_2 and thus also for D_1. We will show that we can modify S into a schedule S' that is still time-feasible for D_2 but violates $S_k' + d_{kl}^{(1)} \leqslant S_l'$, thus obtaining a contradiction. To this end, we apply lemma 3.2 to α_k with $d > S_l - S_k - d_{kl}^{(1)}$ if $d_{kl}^{(2)} = -\infty$, and $d = S_l - S_k - d_{kl}^{(2)}$ otherwise. In both cases we obtain that $S_l' = \max\{S_l, S_k + d_{kl}^{(2)} + d\} = S_l$. So in the first case, we obtain the direct contradiction

$$S_l' - S_k' = S_l - S_k - d < d_{kl}^{(1)},$$

while in the second case, we obtain $S_k' = S_l - d_{kl}^{(2)}$ and thus $S_k' + d_{kl}^{(1)} > S_k' + d_{kl}^{(2)} = S_l = S_l'$, again a contradiction.

The other statements follow from the fact that $ES_{D+\Theta_i} = (d_{11}^{(i)}, \ldots, d_{1n}^{(i)})$ and the monotonicity of κ. □

This monotonicity property implies that in order to determine the optimum value $\rho(\kappa; x)$ for any cost function κ, one need only consider the $\subseteq$-minimal partial orders in $\mathcal{F}$, or, equivalently, the minimal feasible distance matrices (with respect to the componentwise ordering). So we obtain the following stronger version of the representation theorem.

3.12. COROLLARY

The optimum value $\rho(\kappa; x)$ of the scheduling problem $P = [A, x, D, \mathcal{N}, \kappa]$ is already obtained as

$$\begin{aligned}\rho(\kappa; x) &= \min\{\kappa(ES_{D+\Theta}; x) \mid \Theta_D \subseteq \Theta, \Theta \text{ minimal feasible}\} \\ &= \min\{\kappa(ES_{D'}; x) \mid D \leqslant D', D' \text{ minimal feasible}\}.\end{aligned}$$

3.4. CONSTRUCTION OF FEASIBLE EXTENSIONS

The structure of $\mathcal{F}$ described in theorem 3.9 and the definition of feasibility give the following characterization of feasible extensions of a problem $P = [A, x, D, \mathcal{N}, \kappa]$.

3.13. LEMMA

An extension Θ of Θ_D is resource-feasible iff, for any $N \in \mathcal{N}$ (or, equivalently, for any $N \in \mathcal{N}_{red}$), there are $\alpha_i, \alpha_j \in N$ with $\alpha_i <_\Theta \alpha_j$.

Proof

Resource-feasibility means that no forbidden set N is an antichain of Θ. Obviously, this is equivalent to the statement that any such set N contains two comparable jobs α_i, α_j. Since each forbidden set contains a forbidden set from $\mathcal{N}_{\text{red}}$, the lemma follows. □

Lemma 3.13 implies that any resource-feasible extension can be constructed from Θ_D by adding successively precedence constraints $\alpha_i < \alpha_j$ to Θ_D until all (non-redundant) forbidden sets are no longer antichains.

This establishes a complete analogy with precedence-constrained scheduling, and shows that the non-redundant forbidden sets $N \in \mathcal{N}_{\text{red}}$ represent the "essential conflicts" arising from the resource constraints. These conflicts are settled by a resource-feasible partial order by adding precedence constraints for each "conflict".

However, the temporal constraints may restrict the ways in which precedence constraints can be added to Θ_D. In theorem 3.9, this is expressed by the fact that $\mathcal{F}_T$ is generally only a convex subset of $\mathcal{O}(A)$, while $\mathcal{F}_R$ is always a filter.

Furthermore, we do not just consider feasible extensions Θ in the representation theorem, but constraint systems of the form $D + \Theta$. Thus instead of computing a feasible extension, it turns out to be better to compute directly the constraint system $D + \Theta$ or, equivalently, its embedded partial order Θ' (which may be different from Θ).

We will first consider the algorithmic treatment of adding a single precedence constraint to a distance matrix.

To this end, let $D' = (d'_{ij})$ be a distance matrix that has a feasible solution (e.g. the distance matrix of a constraint system $D + \Theta$ in the sense of (3.6)). We say that the precedence constraint $\alpha_k < \alpha_l$ may be *added* to D' if also the constraint system

$$(3.11) \qquad \begin{cases} S_i + d'_{ij} \leqslant S_j & \text{for all } i \neq j \\ S_k + x_k \leqslant S_l \end{cases}$$

has a solution.

3.14. LEMMA

In the above situation, $\alpha_k < \alpha_l$ *may be added to* D' *iff*

$$x_k \leqslant -d'_{lk}. \tag{3.12}$$

In that case, the distance matrix $D'' = (d''_{ij})$ *representing the constraint system* (3.11) *is obtained in* $O(n^2)$ *time as*

$$d''_{ij} = \max\{d'_{ij},\ d'_{ik} + x_k + d'_{lj}\} \tag{3.13}$$

Proof

Because of proposition 2.1 and corollary 2.3, the d'_{ij} represent lengths of longest paths in some digraph G'. Adding the constraint $\alpha_k < \alpha_l$ then means adding the edge (α_k, α_l) with length x_k to G', and the resulting graph G'' represents the system (3.11). This system has a solution iff adding $\alpha_k < \alpha_l$ does not create a cycle of positive length in G''. Since any newly created cycle in G'' must contain the edge (α_k, α_l), the longest such cycle will consist of the edge (α_k, α_l) and a longest path from α_l to α_k in G', which has length d'_{lk}. So the length of the longest newly created cycle is $x_k + d'_{lk}$, which gives (3.12).

If there is no cycle of positive length in G'', then the longest path lengths d''_{ij} of G'' are obviously obtained by comparing the old path lengths d'_{ij} in G' with the possibly newly created path $\alpha_i - \alpha_k - \alpha_l - \alpha_j$ of length $d'_{ik} + x_k + d'_{lj}$. This gives (3.13). □

Of course, adding a precedence constraint $\alpha_k < \alpha_l$ to D' is only meaningful if it is not already contained in the embedded partial order Θ' of D'. If already $\alpha_k <_{\Theta'} \alpha_l$, then $x_k \leqslant d'_{kl}$ because of theorem 3.1, and thus

$$d'_{ik} + x_k + d'_{lj} \leqslant d'_{ik} + d'_{kl} + d'_{lj} \leqslant d'_{ij}$$

because of corollary 2.3 (1). Hence (3.13) just reduces to $d''_{ij} = d'_{ij}$ for all i, j. In that case, also (3.12) is satisfied since $x_k \leqslant d'_{kl}$ and $d'_{kl} + d'_{lk} \leqslant 0$.

We will use the notation $D'' = D' + [\alpha_k < \alpha_l]$ to express that D'' arises from D' by adding $\alpha_k < \alpha_l$.

The algorithmic approach based on lemmas 3.13 and 3.14 constructs distance matrices rather than (feasible) extensions. Of course, each constructed distance matrix is associated with its embedded partial order, and so one may also express the construction process as a construction of partial orders. The main point here is that different partial orders Θ_1, Θ_2 may induce the same constraint system $D + \Theta_1 = D + \Theta_2$. However, if we consider the unique distance matrix D' representing this constraint system (in the sense of proposition 2.2), then D' has a uniquely determined embedded partial order Θ', and $D + \Theta' = D + \Theta_1 = D + \Theta_2$, while Θ' may be different from Θ_1 and Θ_2. So restricting ourselves to the construction of distance matrices may be interpreted as constructing only "canonical" partial orders Θ' that represents a constraint system $D + \Theta$. (In fact,

Θ' is the least upper bound in the semi-lattice $\mathcal{O}(A)$ of all partial orders Θ with $D + \Theta = D + \Theta'$.)

Assume that the initial distance matrix D, the embedded partial order Θ_D, and the system $\mathcal{N}_{\text{red}}$ of minimal forbidden antichains of Θ_D are given. Suppose further that Θ is a feasible extension of Θ_D. We then want to construct the distance matrix D' representing the (solvable) constraint system $D + \Theta$.

To this end, let $\alpha_{i_1} < \alpha_{j_1}, \ldots, \alpha_{i_k} < \alpha_{j_k}$ be the additional precedence constraints of Θ w.r.t Θ_D, i.e. $\Theta \setminus \Theta_D = \{\alpha_{i_1} < \alpha_{j_1}, \ldots, \alpha_{i_k} < \alpha_{j_k}\}$. Then all systems $D + \{\alpha_{i_r} < \alpha_{j_r} \mid r = 1, \ldots, s\}$, $s = 1, \ldots, k$, have a feasible solution, and the associated distance matrices $D_1, \ldots, D_k$ for these systems are obtained iteratively by computing D_s from D_{s-1} (with $D_o = D$) as $D_s = D_{s-1} + [\alpha_{i_s} < \alpha_{j_s}]$ in the sense of lemma 3.11.

Note that some of the D_s may be equal to D_{s-1}. This is the case if $x_{i_s} \leqslant d_{i_s j_s}^{(s-1)}$. This condition can be easily checked before performing an iteration.

For the last distance matrix D_k in this sequence, we have:

3.15. LEMMA

D_k is the distance matrix of the constraint system $D + \Theta$.

Proof

Let D' be the distance matrix of $D + \Theta$, and let Θ_k denote the embedded partial order of D_k. By construction, each D_s is constructed in a "minimal" way from D_{s-1} in order to fulfil the additional constraint $\alpha_{i_s} < \alpha_{j_s}$ given by Θ. This obviously implies that $D_k < D'$.

In the converse direction, observe that $\Theta \subseteq \Theta_k$ by construction of D_k, and so $D + \Theta \leqslant D + \Theta_k$ because of lemma 3.11. Since D_k is the distance matrix of $D + \Theta_k$, we obtain $D' \leqslant D_k$. □

Combining corollary 3.12, lemma 3.13 and lemma 3.14, one obtains that, in order to construct (minimal) feasible distance matrices, it suffices to consider distance matrices constructed from D in the sense of lemma 3.14 by adding precedence constraints for the forbidden sets in the sense of lemma 3.13. So we have:

3.16. COROLLARY

The optimum value $\rho(\kappa, x)$ of the scheduling problem $P = [A, x, D, \mathcal{N}, \kappa]$ is obtained as

$$\rho(\kappa, x) = \min\{\kappa(ES_{D'};\ x) \mid D' = D + [\alpha_{i_1} < \alpha_{j_1}] + \cdots + [\alpha_{i_k} < \alpha_{j_k}]$$

for all choices $\alpha_{i_s} < \alpha_{j_s}$ of precedence constraints such that each forbidden set $N \in \mathcal{N}$ contains a pair $\{\alpha_{i_s}, \alpha_{j_s}\}$}.

This corollary is the starting point for the branch-and-bound algorithm in sect. 4.

3.4. THE GEOMETRIC STRUCTURE OF THE SET OF FEASIBLE SCHEDULES

As a side result of the main representation theorem, we also obtain a geometric characterization of the set $\mathcal{S}$ of all feasible schedules viewed as subset of $\mathbb{R}^n$.

3.17. THEOREM

Let $P = [A, x, D, \mathcal{N}, \kappa]$ be a scheduling problem. Then the set $\mathcal{S}$ of feasible schedules for P has a representation as

$$\mathcal{S} = \mathcal{S}_{\Theta_1} \cup \cdots \cup \mathcal{S}_{\Theta_k},$$

where $\Theta_1, \ldots, \Theta_k$ denote the minimal feasible extensions of Θ_D, and $\mathcal{S}_{\Theta_s}$ is the convex polyhedron of all schedules solving the constraint system $D + \Theta_s$ ($s = 1, \ldots, k$).

Proof

This follows immediately from lemmas 3.6, 3.7, and 3.11. □

In other words, $\mathcal{S}$ is the union of (not necessarily disjoint) convex polyhedrons, each of which represents the set of solutions of a system of linear inequalities of the form (3.6), i.e. with common inequalities $S_i + d_{ij} \leqslant S_j$ for all d_{ij} ($i \neq j$), and with additional inequalities from Θ_s.

Each such polyhedron $\mathcal{S}_{\Theta_s}$ contains a unique componentwise minimal point viz. $ES_{D+\Theta_s}$, and so any non-decreasing function of the S_i (or, equivalently, of the completion times $C_i = S_i + x_i$) attains its minimum on one of these points $ES_{D+\Theta_s}$.

The main representation theorem may then be regarded as establishing a *1–1 correspondence* between the geometrically minimal points and certain partial orders (the minimal feasible extensions). Furthermore, the construction of a feasible partial order is equivalent to constructing a feasible point of $\mathcal{S}$ from the outside (more precisely, from a point of the larger polyhedron $\mathcal{P}$ given by the inequalities of D) by introducing successively new inequalities of the form $S_i + x_i \leqslant S_j$ for the forbidden sets.

Though this geometric point of view is still only descriptive, it might become the starting point for polyhedral methods for the solution of general scheduling problems.

The mentioned 1–1 correspondence between minimal points of $\mathcal{S}$ and minimal feasible extensions of Θ_D might suggest that the standard representation given by the distance matrix D and the set of essential conflicts $\mathcal{N}_{\text{red}}$ (the system of all $\subseteq$-minimal forbidden antichains of the embedded partial order Θ_D) is a *unique discrete description* of the feasible domain $\mathcal{S}$ of a scheduling problem.

For precedence-constrained scheduling, this is indeed the case [23,24,18]. In the general case treated here, however, one can only give a unique description by a derived distance matrix and system of forbidden sets which may be different from D and $\mathcal{N}_{\text{red}}$. The reason for this is again the involved interaction of temporal constraints and resource constraints.

Suppose that $\mathcal{S}$ is the set of feasible schedules for a scheduling problem P over $A = \{\alpha_1, \ldots, \alpha_n\}$ with processing time vector $x = (x_1, \ldots, x_n)$. From $\mathcal{S}$, we

can in a natural way construct a matrix $D_{\mathcal{S}} = (d_{ij}^{\mathcal{S}})$ and a set system $\mathcal{N}_{\mathcal{S}}$ by putting

$$(3.14) \quad d_{ij}^{\mathcal{S}} := \sup\{t \in \mathbb{R}^1 \mid S_i + t \leqslant S_j \text{ for all } S \in \mathcal{S}\},$$

where $\sup \emptyset = -\infty$, and

$$(3.15) \quad N \in \mathcal{N}^{\mathcal{S}} :\Leftrightarrow \begin{cases} \text{no } S \in \mathcal{S} \text{ schedules } N \text{ in parallel,} \\ \text{but each proper subset of } N \text{ is} \\ \text{scheduled in parallel by some } S \in \mathcal{S}, \\ \text{and no pair } \alpha_i, \alpha_j \in N \text{ fulfils } S_i + x_i \leqslant S_j \\ \text{for all } S \in \mathcal{S}. \end{cases}$$

So $N \in \mathcal{N}^{\mathcal{S}}$ if it is a candidate set for parallel scheduling (there is no precedence constraint $\alpha_i < \alpha_j$ on N), but only all proper subset are actually scheduled in parallel.

It is easy to see that $D_{\mathcal{S}}$ fulfils $d_{ij}^{\mathcal{S}} + d_{jk}^{\mathcal{S}} \leqslant d_{ik}^{\mathcal{S}}$ for all i, j, k. Hence $D^{\mathcal{S}}$ is a distance matrix (i.e. a matrix of longest path lengths). We call $D^{\mathcal{S}}$ and $\mathcal{N}^{\mathcal{S}}$ the *derived distance matrix* and *derived system of forbidden sets* of P, respectively. One then obtains easily:

3.18. LEMMA

Let $\mathcal{S}_1$, $\mathcal{S}_2$ be the feasible domain of two scheduling problems over A and x, and let $D_i^{\mathcal{S}}$ and $\mathcal{N}_i^{\mathcal{S}}$ $(i = 1, 2)$ be the derived distance matrix and system of forbidden sts. Then $\mathcal{S}_1 = \mathcal{S}_2$ iff $D_1^{\mathcal{S}} = D_2^{\mathcal{S}}$ and $\mathcal{N}_1^{\mathcal{S}} = \mathcal{N}_2^{\mathcal{S}}$.

In precedence-constrained scheduling, one obtains the stronger result that, if $\mathcal{S}_1 = \mathcal{S}_2$, then equality holds also for the standard representation, i.e. $D_1 = D_2$ and $\mathcal{N}_{\text{red}}^1 = \mathcal{N}_{\text{red}}^2$. This characterization property of "essentially distinct" problems is, however, lost for general temporal constraints. This will be demonstrated in the next example

3.19. EXAMPLE

Let $P = [A, x, D, \mathcal{N}]$ be given by

$$A = \{\alpha_1, \alpha_2, \alpha_3\}, \quad x = (2, 1, 1),$$

$$D = \begin{pmatrix} 0, & 1, & 1 \\ -\infty, & 0, & 0 \\ -\infty, & 0, & 0 \end{pmatrix}, \quad \mathcal{N}_{\text{red}} = \{\{\alpha_1, \alpha_2, \alpha_3\}\}.$$

Then every feasible schedule must schedule α_2 and α_3 completely in parallel at any time after the completion of α_1, i.e.

$$\mathcal{S} = \{(S_1, S_2, S_3) \in \mathbb{R}^3_{\geqslant} \mid S_1 + 2 \leqslant S_2,\ S_1 + 2 \leqslant S_3,\ S_2 = S_3\}.$$

Hence

$$D^{\mathcal{S}} = \begin{pmatrix} 0, & 2, & 2 \\ -\infty, & 0, & 0 \\ -\infty, & 0, & 0 \end{pmatrix} \text{ and } \mathcal{N}^{\mathcal{S}} = \emptyset.$$

The reason is that, although for each i, j there is a time-feasible schedule S with $S + d_{ij} = S_j$, there may not be a feasible such schedule, (the pair 1, 2 in the example) and, furthermore, that to a proper subset of some $N \in \mathcal{N}_{\text{red}}$, there may not be a feasible schedule that schedules N in parallel (the set $\{\alpha_1, \alpha_2\}$ in the example). Both properties are valid for precedence constrained scheduling and establish the mentioned characterization of essentially distinct problems.

3.5. REMARKS ON STABILITY AND SENSITIVITY ANALYSIS

We will now investigate the sensitivity behaviour of the optimum value and of optimal feasible extensions with respect to a variation of the processing times. Again, the structural approach taken in this paper turns out to provide a very natural access to such questions.

A basic difference with precedence-constrained scheduling is given by the fact that the optimum value $\rho(\kappa; x)$ may not exist for some x, even if the temporal constraints given by D are solvable. An additional complication arises from the dependence of the distance matrix D and thus the feasibility domain $\mathcal{S}$ from the processing times, something which is inherent to time window constraints and establishes another basic difference with precedence-constrained scheduling.

We will first study the effect of a variation of the processing times on the distance matrix D. This effect is caused by the transformation rules (2.4)–(2.7) which show that each arc length l_{ij} of the underlying digraph G is an (affine) linear function of x_i and x_j, i.e. $l_{ij} = a_{ij} + b_{ij}x_i + c_{ij}x_j$ with $a_{ij} \in \mathbb{R}^1$ and $b_{ij}, c_{ij} \in \{-1, 0, +1\}$.

Note that all results of this section remain valid if we assume that the temporal constraints are given by a digraph G whose arc lengths $l_{ij}(x)$ are arbitrary (affine) linear functions of the processing times.

A system of such temporal constraints has a solution iff G contains no cycle of positive length w.r.t. the processing time $x_1, \ldots, x_n$. Let X_T denote the set of all $x \in \mathbb{R}^n_{\geqslant}$ for which the system of temporal constraints is solvable.

3.20. LEMMA

(1) *X_T is a convex polyhedron.*

(2) *For any $x \in X_T$, every entry $d_{ij}(x)$ of the associated distance matrix $D(x)$ is a piecewise linear and convex function in every variable x_i.*

(3) *The associated embedded partial orders $\Theta_{D(x)} = (A, <_x)$, $x \in X_T$, have in $\mathcal{O}(A)$ a greatest lower bound $\Theta_T = (A, <_T)$, which is given by*

$$\alpha_i <_T \alpha_j \Leftrightarrow \alpha_i <_x \alpha_j \quad \text{for all } x \in X_T. \tag{3.16}$$

Proof

The set X_T is obviously the set of solutions of the system of *cycle inequalities*

$$\sum_{(i,j) \in C} l_{ij}(x) \leqslant 0 \tag{3.17}$$

given by the directed cycles C of G. Since each $l_{ij}(x)$ is linear in x, this is a system of linear inequalities and (1) follows.

(2) follows from the fact that $d_{ij}(x)$ is the maximum of path lengths, and that each path length is a linear function of x.

The partial order Θ_T is the intersection of all $\Theta_{D(x)}$, and hence their greatest lower bound in $\mathcal{O}(A)$. □

The partial order Θ_T may be regarded as representing the "real" *time-independent* precedence constraints, while each embedded partial order $\Theta_{D(x)}$ may contain additional precedence constraints caused by sufficiently short processing times (i.e. $x_i \leqslant d_{ij}(x)$ in the sense of theorem 3.1).

In particular, in precedence-constrained scheduling, Θ_T is just the initially given partial order of precedence constraints.

So for each $x \in X_T$, one obtains a scheduling problem $[A, x, D(x), \mathcal{N}, \kappa]$, where $\mathcal{N}$ is always the same system of forbidden sets and $D(x)$ is the distance matrix representing the temporal constraints given by the digraph G with arc lengths $l_{ij}(x)$. Since $D(x)$ and the embedded partial order $\Theta_{D(x)}$ also depend on x, the set of associated feasible extensions of $\Theta_{D(x)}$ will in general also depend on x.

However, there is a different behaviour between resource-feasible and time-feasible extensions. To see this, note first that all extensions of some $\Theta_{D(x)}$ are also extensions of Θ_T, the time-independent partial order contained in all $\Theta_{D(x)}$. So for a particular $x \in X_T$, the sets $\mathcal{F}_T(x)$, $\mathcal{F}_R(x)$ and $\mathcal{F}(x)$ of all time-feasible, resource-feasible and feasible extensions of Θ_T are well-defined. We then easily obtain:

3.21. LEMMA

For each $x \in X_T$, $\mathcal{F}(x) = \mathcal{F}_T(x) \cap \mathcal{F}_R$, *where* $\mathcal{F}_R = \mathcal{F}_R(x) = \{\Theta \supseteq \Theta_T \mid$ *no antichain of* Θ *contains a forbidden set* $N \in \mathcal{N}\}$.

In other words, the change in the set $\mathcal{F}(x)$ of feasible extensions that occurs from varying the processing times is entirely caused by time-feasibility (and not by resource-feasibility). This is a major drawback in comparison with precedence constrained scheduling, where $\mathcal{F}(x)$ never changes ($\mathcal{F}(x) = \mathcal{F}_R$ for all x), and causes the nasty behaviour illustrated below in example 3.23.

An extension Θ of Θ_T is by definition time-feasible for a given $x \in X_T$, if the constraint system $D(x) + \Theta$ is solvable. Let, for fixed Θ, X_Θ denote the set of all processing time vectors x for which $D(x) + \Theta$ is time-feasible. We then have:

3.22. LEMMA

For each extension Θ *of* Θ_T, X_Θ *is a (possibly empty) convex polyhedron.*
Furthermore, $\mathcal{F}_T(x) = \{\Theta \supseteq \Theta_T \mid X_\Theta \neq \emptyset\}$, *and* $\mathcal{F}(x) = \{\Theta \in \mathcal{F}_R \mid X_\Theta \neq \emptyset\}$.

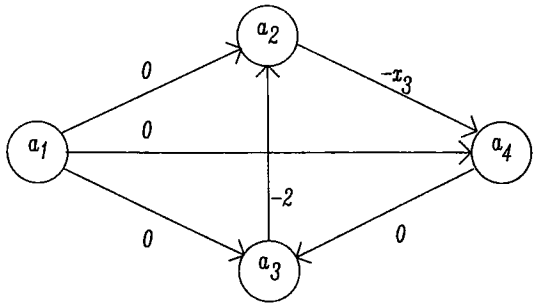

Fig. 6. Temporal constraint graph G.

Proof

Obvious from lemma 3.20 and the definition of time-feasibility. □

3.22. EXAMPLE

Consider a scheduling problem with 4 jobs, i.e. $A = \{\alpha_1, \ldots, \alpha_4\}$ and temporal constraints given by the following time lags:

- minimum start-to-start lags $S_1 \leqslant S_2$, $S_1 \leqslant S_3$, $S_1 \leqslant S_4$, $S_3 \leqslant S_4$
- the minimum start-to-finish lag $S_2 \leqslant S_3 + x_3$
- the maximum start-to-start lag $S_4 \leqslant S_2 + 2$.

These temporal constraints result in the digraph given in fig. 6. So $X_T = \mathbb{R}^4_{\geqslant}$, i.e. the temporal constraints are solvable for any vector of processing times.

Furthermore, $\Theta_{D(x)} = \Theta_T$ is the degenerate order on A (all 4 jobs are pairwise unrelated).

Now let $\mathcal{N} = \{\{\alpha_2, \alpha_4\}\}$. Then $\mathcal{F}_R = \{\Theta_1, \Theta_2\}$, where of course $\Theta_1 = (A, \alpha_2 < \alpha_4)$ and $\Theta_2 = (A, \alpha_4 < \alpha_2)$. The associated constraint systems $D(x) + \Theta_i$ are obtained from the graph G by adding arcs (α_2, α_4) with length $l_{24} = x_2$ and changing $l_{42} = -2$ into $l_{42} = x_4$, respectively.

The cycle inequalities in the modified graphs then give

$$X_{\Theta_1} = \left\{x \in \mathbb{R}^4_{\geqslant} \middle| x_2 \leqslant 2\right\}$$

and

$$X_{\Theta_2} = \left\{x \in \mathbb{R}^4_{\geqslant} \middle| x_4 \leqslant x_3\right\}.$$

So if we fix $x_1 = c_1$ and $x_4 = c_4$, we obtain the situation in the $x_2 - x_3$-plane depicted in fig. 7.

Consider the line segment $L = x^1x^2$ in the $x_2 - x_3$-plane. For all $x \in L$ with $x_2 \leqslant 2$, Θ_1 is the only optimal extension, for all $x \in L$ with $x_2 > 2$ and $x_3 < c_4$ there is no optimal extension, and for $x \in L$ with $x_2 > 2$ and $x_3 \geqslant c_4$ the optimal extension is given by Θ_2.

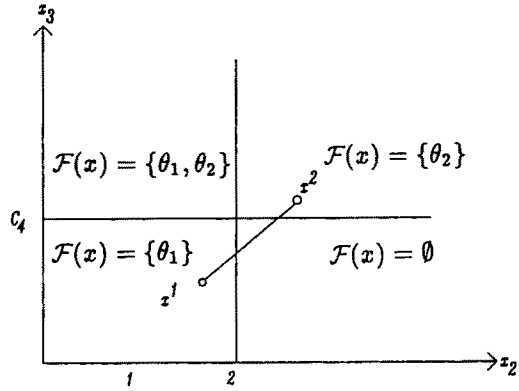

Fig. 7. Systems of feasible extensions.

For the respective *ES*-schedules and associated cost values one obtains

$$\kappa(ES_{D+\Theta_1};\ x) = \kappa(c_1,\ x_2,\ x_3,\ x_2 + c_4),$$

$$\kappa(ES_{D+\Theta_2};\ x) = \kappa(c_1,\ x_2 + c_4,\ x_3,\ c_4).$$

Since both argument vectors are incomparable (with respect to the componentwise ordering on $\mathbb{R}^4$), there are cost functions with $\kappa(c_1,\ x_2,\ x_3,\ x_2 + c_4) > \kappa(c_1,\ x_2 + c_4,\ x_3,\ c_4)$.

This shows that, even for processing times x that are monotonically increasing along a line segment, the optimum value $\rho(\kappa;\ x)$ may decrease and even not be defined at all. Even more, as variation of x_4 shows, this may happen in an arbitrarily small neighborhood of some vector x_0. All this shows:

3.23. REMARK

In general, the domain X_T of the optimum value function $\rho(\kappa;\ \cdot)$ is not convex, and $\rho(\kappa;\ \cdot)$ is not monotonically increasing in the processing times.

So positive results may only expected when the occurence of cycles is restricted. A strong such "restricting" assumption is the basis of the following theorem.

3.24. THEOREM

Let, in the digraph of temporal constraints, all $l_{ij}(x) \geqslant 0$ for any $x \in \mathbb{R}^n_>$ and $X_T \neq \emptyset$.

Then $X_T = \mathbb{R}^n_>$ and $\rho(\kappa;\ \cdot)$ is monotonically increasing in the processing times x.

Moreover, $\rho(\kappa;\ \cdot)$ is (uniformly) continuous if κ is, and $\rho(\kappa_j;\ x^j) \to \rho(\kappa;\ x)$ for $j \to \infty$ whenever $\kappa_j \to \kappa$ uniformly and $x^j \to x$ $(j \to \infty)$.

Proof

Since all $l_{ij}(x) \geqslant 0$, and since adding precedence constraints preserves this property, one obtains that $\mathcal{F}(x)$ is independent of x, and that each constraint system $D(x) + \Theta$, $\Theta \in \mathcal{F}$, is solvable for any $x \in \mathbb{R}^n_>$. Hence $X_T = \mathbb{R}^n_>$.

The property $l_{ij}(x) \geq 0$ for all x implies that all x_i in the linear function have non-negative coefficients, and thus all $l_{ij}(x)$ are monotonically increasing in x. So also all constraint systems $D(x) + \Theta$, $\Theta \in \mathscr{F}$, and hence all $ES_{D(x)+\Theta}$, $\Theta \in \mathscr{F}$, are monotonically increasing in x.

This obviously implies the monotonicity property. The other statements follow from $\mathscr{F}(x) = \mathscr{F}$ for all x and the main representation theorem. □

Theorem 3.24 obviously covers the case of precedence-constrained scheduling, where each $l_{ij}(x)$ is of the form $l_{ij}(x) = x_i$.

Note, however, that it does not suffice to assume that the digraph representing the temporal constraints is acyclic. Then arc lengths $l_{ij}(x)$ may be negative and cycles may be created by adding precedence constraints on forbidden sets (which can be forced by temporal constraints to be unique on a forbidden set). So essentially (i.e. by adding some additional jobs to model the forcing), the same situation as in example 3.22 can be constructed.

With respect to the resource restrictions and the cost function, the monotonicity behaviour of the optimum value is much better, as the final result of this section shows:

3.25. THEOREM

Let $P_1 = [A, x, D, \mathscr{N}_1, \kappa_1]$ *and* $P_2 = [A, x, D, \mathscr{N}_2, \kappa_2]$ *be scheduling problems with the same jobs and temporal constraints. Let* $\rho_i(\kappa_i; x)$ $(i = 1, 2)$ *denote the associated optimum values.*

Then $\rho_1(\kappa_1; x) \leq \rho_2(\kappa_2; x)$ *if* $\kappa_1 \leq \kappa_2$ *and each* $N \in \mathscr{N}_2$ *is contained in some* $N' \in \mathscr{N}_1$.

Proof

The assumption on $\mathscr{N}_1$ and $\mathscr{N}_2$ implies that $\mathscr{F}_R^1 \subseteq \mathscr{F}_R^2$ for the respective domains of resource-feasible extensions of Θ_D. Combining this with theorem 3.9 and theorem 3.8 proves the result. □

The assumption of $\mathscr{N}_1$ and $\mathscr{N}_2$ is e.g. fulfilled if both problems have the same resource types, but with smaller requirements per job in P_1 and/or larger availabilities in P_2.

4. Algorithmic aspects

4.1. THE ALGORITHM

As mentioned before, the algorithm based on the structural approach taken in this paper is a branch-and-bound algorithm that constructs feasible extensions Θ

(or rather, the associated distance matrix of $D + \Theta$) according to the results of sect. 3.3.

So a *node* in the underlying *search tree* is a distance matrix $D' \geqslant D$ that represents the original temporal constraints given by D and some additional precedence constraints introduced on certain forbidden sets.

D' is feasible if all forbidden sets $N \in \mathcal{N}$ are *destroyed*, i.e. if D' contains a precedence constraint $\alpha_i < \beta_j$ (which is equivalent to $x_i \leqslant d'_{ij}$ by theorem 3.1) for each $N \in \mathcal{N}$. Then D' represents a feasible solution with objective value $\kappa(ES_{D'}; x)$. This value will be compared with the currently best value ρ^* (and substituted for it if it is smaller). So ρ^* always represents the best value so far obtained.

If D' is not feasible, then there will be information available on the yet undestroyed forbidden sets and on the precedence constraints that may destroy them. This information can e.g. be handled by maintaining a list of precedence contraints "$\alpha_i < \alpha_j$" contained in not yet destroyed forbidden sets, possibly together with a globally available (static) array representing which precedence constraint destroys which forbidden set in order to test this on $O(1)$ time.

Then branching from D' is based on this information about not yet destroyed forbidden sets and precedence constraints destroying them. There are two principle ways for organizing the branching.

The first way takes a not yet destroyed forbidden set N_o (possibly the "next" in a predetermined order on the forbidden sets, or with respect to some dynamic rule involving e.g. increases in the associated ES-schedules or cost values) and all precedence constraints that destroy N_o and may be added to D' (in the sense of lemma 3.14). Each such precedence constraint $\alpha_i < \alpha_j$ then represents a *branching* to a subset of the feasibility domain $\mathcal{F}$ in which feasible schedules fulfil the additional constraint $S_i + x_i \leqslant S_j$.

The second way is a variation in which one considers all precedence constraints that may be added to D' and destroy some forbidden set (instead of a fixed set N_o as above), possibly again supported by static or dynamic rules of choice. This way was used in the computations reported on in sect. 4.2.

Note that if there is no precedence constraint that may be added to D', then there is no path from D' to a feasible solution and no branching is necessary in the current node.

As *lower bound LB* on $\rho(\kappa; x)$ in the current node with distance matrix D' we take the associated cost value $\kappa(ES_{D'}; x)$. This is a natural lower bound because of lemma 3.11, though possibly not the best one. Better lower bounds may be derived for special problems (e.g., in job shop scheduling, lower bounds based on 1-machine problems have led in [7] to a solution of the famous 10×10-job shop scheduling problem), or by deriving bounds from the system of yet undestroyed sets, by subgradient methods or other standard techniques. So far, our algorithmic experience has only been based on the "natural" lower bound.

The *selection rule* we have used was depth first search (DFS). This can of

course be replaced by any other standard rule for selecting the next branch in the search tree, though DFS compares favourably with others because of the data handling.

So in all variations of this branch-and-bound scheme, a selected branch is associated with a precedence constraint $\alpha_i < \alpha_j$ that leads to the new distance matrix $D'' = D' + [\alpha_i < \alpha_j]$ that is computed from D' in $O(n)$ time according to (3.10).

The branch is *pruned* if the lower bound LB associated with D'' ($LB = \kappa(ES_{D''};\ x)$ in our computations) exceeds the currently best value ρ^*. Note that in order to determine the special lower bound $\kappa(ES_{D''};\ x)$, it suffices to compute only the values $(ES_{D''})_j$ that are needed as arguments in the cost function κ (possibly only one as for project duration when there is a last job representing the end of the project). So this lower bound can be obtained in $O(n)$ time without computing the entire matrix D''.

These are the basic ingredients of the branch and procedure which essentially consist in an "on-line" settling of conflicts on forbidden sets by introducing precedence constraints on them and by calculating the new distance matrix. So the forbidden sets and the distance matrix determine the "size" of the problem in computational respect.

Here again, a reduction to the "computational core" of the problem is possible. In order to keep $|\mathcal{N}|$ small, restriction to the system $\mathcal{N}_{\text{red}}$ of *reduced forbidden sets* ($\subseteq$-minimal forbidden antichains of Θ_D, cf. sect. 3.1) is very reasonable, though it takes some additional effort. Methods to carry out this reduction have been developed in [2]. Compared with the time spent on the branch-and-bound part of the algorithm, the time needed to compute $\mathcal{N}_{\text{red}}$ is neglectably small.

Besides the size of $\mathcal{N}$, also the size of D can be kept small. To this end, call a job α_i *resource-essential* if $\alpha_i \in N$ for some $N \in \mathcal{N}_{\text{red}}$ (i.e. if it needs a "scarce" resource), and *cost-essential* if the completion time C_i of α_i is an essential variable of κ (i.e. the values of C_i matter).

4.1. LEMMA

The data required in the distance matrix D' for updating, checking for embedded and addable precedence constraints, and for computing the associated cost $\kappa(ES_{D'};\ x)$ consists of those d'_{ij}, for which α_i is resource-essential or $i = 1$, and α_j is resource- or cost-essential.

Proof

Checking for embedded and addable precedence constraints $\alpha_i < \alpha_j$ requires only resource-essential jobs because of theorem 3.1 and (3.12). Since precedence constraints are only added between resource-essential jobs, updating d'_{ij} after adding $\alpha_k < \alpha_l$ requires according to (3.13) only knowledge of entries d'_{ij}, d'_{ik} and d'_{kj} which all belong to resource-essential jobs.

The calculation of the cost value requires knowledge of the d'_{1j} for which α_j is cost-essential. Their updating after adding $\alpha_k < \alpha_l$ requires knowledge of d'_{1j}, d'_{1k}, and d'_{kj}, i.e. of entries for which the row (other than 1) corresponds to a resource-essential job and the column to a resource- or cost-essential job. □

Note that additional reduction tricks, such as representing a term of the form $\max\{C_{i_1}, \ldots, C_{i_k}\}$ in the cost function by the completion time of a dummy job α_{n+1} and precedence constraints $\alpha_{i_s} < \alpha_{n+1}$, $s = 1, \ldots, k$, may help to reduce the really needed part of the distance matrix even further.

Altogether this shows that the computational core is given by the set of cost-essential and resource-essential jobs which may be small compared with the set of all jobs in many applications. Only the processing step of determining $\mathcal{N}_{\text{red}}$ and D requires knowledge of the entire data; afterwards, the reduced data set suffices for all computations.

As with the transformation to the standard representation, the reduction to cost- and resource-essential jobs may be carried out internally, i.e. completely hidden from the user, which is e.g. particularly useful in decision support systems.

4.2. COMPUTATIONAL EXPERIENCE

Computational experience with this approach for the purely precedence-constrained case has already been reported in [23,24] and the master thesis [17]. In particular, [17] contains a detailed comparison with the integer programming approach taken in [31]. The two approaches were compared on examples partly taken from the literature ranging from 10 to 31 jobs and from 12 to 440 reduced forbidden sets and covering various applications from civil engineering (complicated precedence constraints) to processor scheduling (simple precedence constraints given by parallel chains of jobs, but much parallelism and many forbidden sets).

The branch-and-bound approach proved to be much superior. The integer programming approach was much slower and in most cases was not able to compute the optimum value within a given time limit (mostly 192 seconds on a Cyber 175), while the branch-and-bound algorithm found the optimum in all cases within 0.073 to 4.278 seconds CPU-time.

The subsequently reported experience in the general case was obtained by M. Bartusch in this thesis [3] on a Cyber 175 at the Rechenzentrum of the RWTH Aachen and, in a different implementation, on an IBM 3081 at the Institute of Operations Research at the University of Bonn.

The first project considered models the construction of a bridge as depicted in Fig. 8.

There are 7 main groups of jobs denoted by A_i (excavation), P_i (ground piling), B_i (concrete foundations), S_i (soil stabilization and concrete framework), M_i (above-grade abutments and piers), T_i (erection of superstructure) and V_i

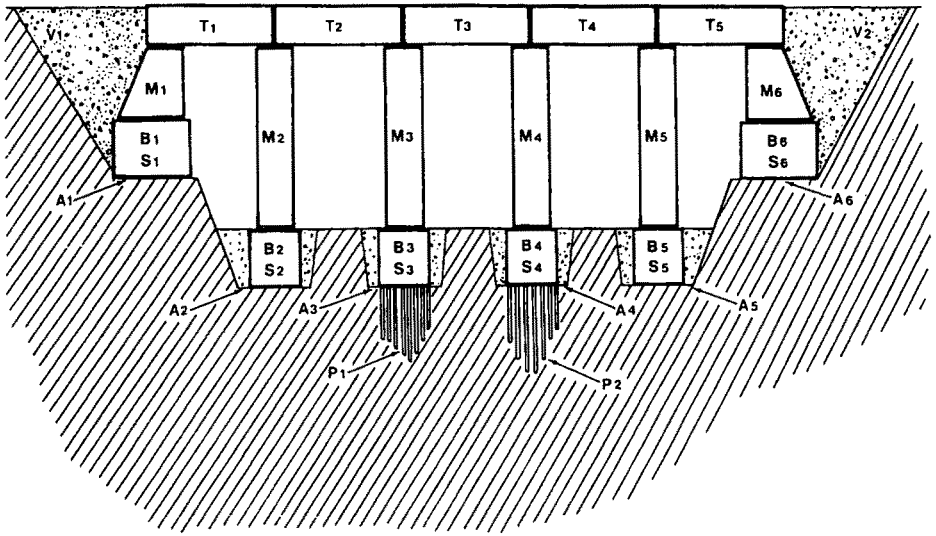

Fig. 8. Bridge construction project.

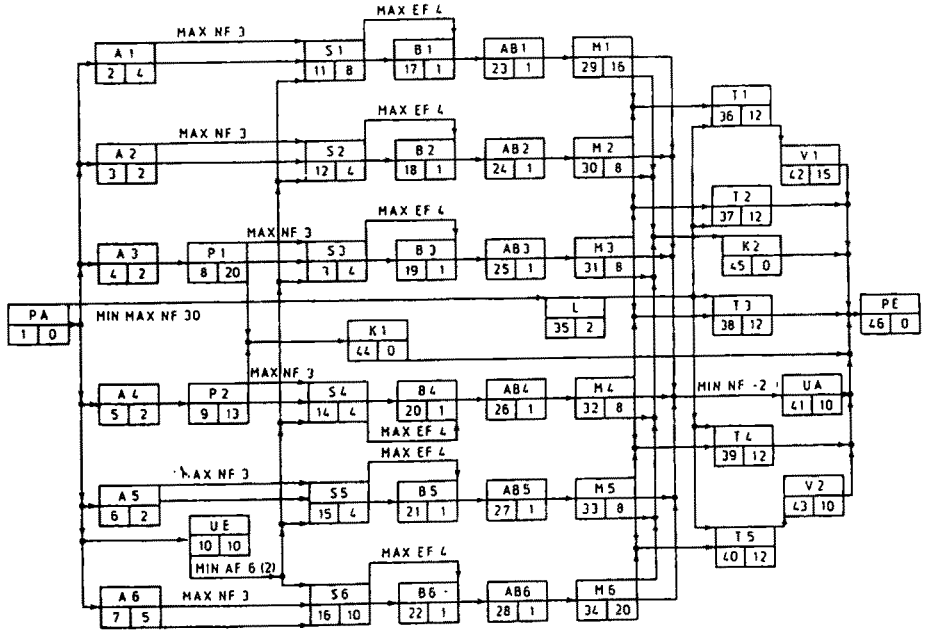

Fig. 9. Temporal constraints of bridge construction project.

(earth fill-up). Some of these jobs are further subdivided, but the basic resource types they require (excavator, pile driver, concrete gang, framework gang, mason gang, crane, bulldozer) are determined by the group in which they belong. The complete temporal constraints are given in fig. 9.

Each of the 46 final jobs α_i is represented by a rectangle that contains its name (with PA and PE denoting project start and end), and, in the second row its index i and its processing time x_i in days. The temporal constraints are given by arrows as in fig. 1, where maximum time lags are indicated by MAX, NF means finish-to-start lag ("Normalfolge"), EF means finish-to-finish lag ("Endfolge"), and absence of any arrow indication means a finish-to-start lag with $l_{ij}^{\min} = 0$ and $l_{ij}^{\max} = \infty$.

The temporal constraints contain several special features required by the civil engineers. For example, due to incoming ground water, there is a maximum time lag of 3 days between the completion of the ground piling in the excavations and the start of erecting the framework for making the concrete foundations. The super structure will be delivered exactly at day 30 after project start, causing a minimum and maximum time lag of 30 days between PA and L (delivery). Also, there is housing for the workers that has to be built up (UE) and closed down (UA) within certain time limits depending on the availability of the different worker gangs, etc.

This problem was first solved for $\kappa = \max$ (overall project duration) and several constellations of available resources, ranging from 1 unit for each of the seven types up to the maximum type required (for which one obtains $\rho(\kappa;\ x) = \kappa(ES_D;\ x) = 60$, i.e. the best value w.r.t. the temporal constraints only).

The actually computed constellations were chosen in such a way that all possible different resource availabilities could be solved by application of the monotonicity result 3.25.

Altogether, 29 different constellations were solved, 15 with 1 scarce resource type, 11 with 2 scarce resource types, and 3 with 3 scarce types. The required CPU-time on the CYBER 175 ranged from 0.2 second to 47.4 seconds for 28 of these problems, but with a large out-runner of 1206.3 for the 29th problem. The average CPU-time was 3.8 seconds for the 28 problems and 45.4 seconds for all problems. On the IBM 3081, the (virtual) running time ranged from 0.02 sec to 62.95 seconds for 27 problems with two outrunners of 213.8 and 400.1 seconds. The "large-outrunner" problem was not solved to optimality within the time-limit of 1800 seconds. The average time (excluding the outrunner) was 27.93 seconds. The number of recursive calls of the branch and bound routine (which corresponds to the number of edges traversed in the decision tree) ranged from 1 to 907,000 with a mean of 48,000.

A variant of this problem (additional start-to-start time lags of minimum duration 2 between UE and S_i, $i = 1, \ldots, 6$; these were due to a modification by the project planner based on a post-optimal analysis of the previous solution) was solved for a typical cost function occcuring in building industry. Besides project

duration, it contained in a weighted sum penalty terms for important individual jobs (K_1, K_2, V_2) that model e.g. the freeing of resources (pile driver) or the end of certain groups of jobs ("subprojects"). These penalties are non-decreasing, piecewise linear functions $f_i(C_i)$ of the completion times C_i of the individual jobs.

The problem was solved for 5 variations of the cost function, with CPU-times ranging from 23.5 sec to 47.3 sec.

Finally, a real-life *pipeline project* [13] with 72 jobs and 14 resource types (of which 7 are scarce) was investigated and solved w.r.t. to overall project duration.

The initial heuristics gave a first upper bound of 136 for the branch-and-bound part of the algorithm which obtained an optimal schedule with length 133 after 13.9 seconds on the CYBER 175 and after 185.6 seconds on the IBM 3081.

It seems that the branch-and-bound approach works very well when the number of reduced forbidden sets is not too large and these sets are small in size. This is usually the case when there are many temporal constraints (which is typical for applications in civil engineering). Here maximum time lags have an additional advantage of possibly pruning parts of the search tree.

On the other hand, when the problem has a "large degree of parallelism", i.e. if there a many (large) forbidden sets or if there are not many temporal constraints (Θ_D is "close" to an antichain), then establishing optimality of a solution will require too much time. This is particularly true for machine scheduling or job shop scheduling problems. For these problems, better lower bounds in the branch-and-bound procedure are very essential, as has been proved by the recent solution of the famous 10×10 job-shop scheduling problem in [7].

Acknowledgement

The last two authors would like to express their appreciation for having had the opportunity of many years of close scientific cooperation with their student M. Bartusch, which lead to his thesis and to the continuation of his work presented in this paper.

References

[1] E. Balas, Project scheduling with resource constraints, in: *Applications of Mathematical Programming*, ed. E.M.L. Beale (The English University Press, London, 1971) 187–200.

[2] M. Bartusch, An algorithm for generating all maximal independent subsets of a poset, Computing 26 (1983) 343–354.

[3] M. Bartusch, Optimierung von Netzplänen mit Anordnungsbeziehungen bei knappen Betriebsmitteln, Thesis, Tech. Univ. of Aachen, 1983.

[4] M. Bartusch, R.H. Möhring and F.J. Radermacher, A conceptional outline of a DSS for scheduling problems in the building industry, Decision Support Systems (1988) to appear.

[5] C. Berge, *Graphs* (North Holland, Amsterdam, 1985).

[6] J. Carlier, Ordonnancements á constraintes disjonctives, RAIRO 12 (1978) 333–351.
[7] J. Carlier and E. Pinson, A branch and bound method for the jobshop problem, preprint, Université de technologie de Compiegne, 1986.
[8] R.W. Conway, W.L. Maxwell and L.W. Miller, *Theory of Scheduling* (Addison-Wesley, Reading, MA, 1967).
[9] M.A.H. Dempster, J.K. Lenstra and A.H.G. Rinnooy Kan, eds. *Deterministic and Stochastic Scheduling* (Reidel, Dordrecht, 1982).
[10] S.E. Elmaghraby, *Activity Networks: Project Planning and Control by Network Models* (Wiley, New York, 1977).
[11] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Freeman, San Francisco, 1979).
[12] M. Glinz and R.H. Möhring, Reduction theorems for networks with general sequencing relations, Methods of Oper. Res. 27 (1977) 124–162.
[13] W. Jurecka, *Netzwerkplanung im Baubetrieb*, Teil 2 (Optimierungsverfahren Bauverlag GmbH, Wiesbaden, 1972).
[14] E.L. Lawler, *Combinatorial Optimization: Networks and Matroids* (Holt, Rinehart and Winston, New York, 1976).
[15] E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan, Recent developments in deterministic sequencing and scheduling: a survey, in: *Deterministic and Stochastic Scheduling*, eds. M.A.H. Dempster et al. (Reidel, Dordrecht, 1982).
[16] J.K. Lenstra and A.H.G. Rinnooy Kan, Complexity of scheduling under precedente constraints, Oper. Res. 26 (1978) 22–35.
[17] G. Mendzigal, Entwurf und Vergleich von Algorithmen zur Optimierung von deterministischen Netzplänen mit Betriebsmittelbeschränkungen, Master Thesis, RWTH Aachen (supervisor: R.H. Möhring), 1984.
[18] R.H. Möhring, Minimizing costs of resource requirements in project networks subject to a fixed completion time, Oper. Res. 32 (1984) 89–120.
[19] R.H. Möhring, Algorithmic aspects of comparability graphs and interval graphs, in: *Graphs and Orders*, ed. I. Rival (Reidel, Dordrecht, 1985) p. 41–101.
[20] R.H. Möhring and F.J. Radermacher, Scheduling problems with resource-duration interaction, Methods of Oper. Res. 48 (1984) 423–452.
[21] K. Neumann, *Operations Research Verfahren*, Band III (Carl Hauser Verlag, München, 1975).
[22] J. Patterson, R. Slowinski, B. Talbot and J. Weglarz, An algorithm for a general class of precedence and resource constrained scheduling problems, 1982, preprint.
[23] F.J. Radermacher, *Kapazitätsoptimierung in Netzplänen*, Math. Syst. in Econ. 40 (Anton Hain, Meisenheim, 1978).
[24] F.J. Radermacher, Scheduling of project networks, Annals of Oper. Res. 4 (1986) 227–252.
[25] F.J. Radermacher, Schedule-induced posets, Discrete Appl. Math 14 (1986) 67–91.
[26] A.H.G. Rinnooy Kan, *Machine Scheduling Problems: Classification, Complexity and Computation* (Nijhoff, The Hague, 1976).
[27] B. Roy, Cheminement et Connexité dans les graphes, Application aux problèmes d'ordonnancement, METRA, Série Spéciale, No. 1, (Thesis), 1962.
[28] B. Roy, Graphes et Ordonnancement, Revue Française de Recherche Opérationelle (1962) 323–333.
[29] J. Schwarze, Netzplantechnik für Praktiker (Verlag Neue Wirtschaftsbriefe, Herne/Berlin, 1974).
[30] R. Seeling, Reihenfolgenprobleme in Netzplänen, Bauwirtschaft (1972) 1897–1904.
[31] F.B. Talbot and J.H. Patterson, An efficient integer programming algorithm with network cuts for solving resource-constrained scheduling problems, Management Sci. 24 (1978) 1136–1174.