Journal of Intelligent Manufacturing (1995) **6**, 145–154

Heuristics for the minimum project-duration problem with minimal and maximal time lags under fixed resource constraints

K. NEUMANN and J. ZHAN

Institut für Wirtschaftstheorie und Operations Research, University of Karlsruhe, D–76128 Karlsruhe, Germany

The authors consider the problem of minimizing the duration of a project under fixed resource constraints. For the case that there are only minimal time lags between the start of successive activities and that therefore the project can be described by an acyclic network, a large number of exact and heuristic algorithms can be found in the literature. In this paper, the authors permit both minimal and maximal time lags between activities. The project can then be modelled by an activity-on-node network containing cycles. Efficient priority-rule heuristics are presented for solving the resource-constrained project-scheduling problem. Computational results are discussed for projects containing up to 1000 activities and several resources.

Keywords: Project scheduling, resource constraints, activity-on-node networks, heuristic procedures, priority-rule methods

1. Introduction

In this paper the following NP-hard problem is dealt with: minimization of the duration of a project subject to limited renewable resources, which are required for carrying out the individual activities of the project. Examples of renewable resources are labour force and machines.

If there are only *minimal time* lags between activities, that is, the project can be modelled by an activity-on-arc network (or CPM network known from the 'critical path method') or an acyclic activity-on-node network, a large number of exact algorithms as well as heuristic procedures have been developed to solve the resource-constrained project-scheduling problem.

The *exact algorithms* are generally of the branch-and-bound type, for example, the algorithms proposed by Talbot and Patterson (1978), Stinson *et al.* (1978), Patterson (1984), Christofides *et al.* (1987), Patterson *et al.* (1989) and Demeulemeester and Herroelen (1992). However, in general, only projects containing up to 50 activities and at most three resources can be treated efficiently using these methods.

Most *heuristics* represent so-called *priority-rule methods*: priorities are assigned to all activities that can be scheduled at a certain point in time or in a time period. The activities are then carried out in the order of decreasing priorities if they cannot be performed simultaneously due to limited resources. This process of scheduling the activities proceeds from time point to time point (or respectively from time period to time period) in chronological order. References for such priority-rule methods are, for example, Davis (1975), Davis and Patterson (1975), Elsayed (1982), and Alvarez-Valdés and Tamarit (1989).

For the case in which there are *minimal and maximal time lags* between activities, that is, CPM networks are no longer applicable and, instead, cyclic activity-on-node networks have to be used to model the projects under consideration, Bartusch (1983) has developed a branch-and-bound method to solve the resource-constrained project-scheduling problem (see also Bartusch *et al.*, 1988). If we truncate the evolution of that algorithm (for example, by omitting some or all backtracking steps), we obtain heuristic procedures. Those heuristics, however, generally require a large computational effort.

Zhan (1991) has investigated how to generalize the priority-rule methods to the case where maximal time lags are present. The heuristics obtained have turned out to be very successful. The procedures generally provide good approximate solutions. For projects with 1000 activities and one resource, the average computing time on a HP workstation of type 9000/360 is 30 s. If we have three resources, the average computation time amounts to 1 min.

2. MPM networks (activity-on-node networks)

The so-called metra-potential method, abbreviated MPM, is a method for temporal analysis of projects, which uses cyclic activity-on-node networks. Thus, the latter project networks are also called *MPM networks.* Before dealing with resource-constrained project scheduling, let us touch upon the basic elements of MPM networks very briefly. The metra-potential method was proposed by Roy (1964); more recent references are Elmaghraby and Kamburowski (1992) and Neumann and Morlock (1993), Section 2.5.

Suppose that the underlying project consists of n activities numbered 1 to n. Activity 1 represents the beginning of the project and activity n its completion. The remaining activities correspond to real activities. Let $D_j \in \mathbf{Z}_+$ be the duration or processing time of activity j (all quantities used in what follows, in particular times and points in time, are assumed to be integers). The dummy activities 1 and n have duration zero: $D_1 := D_n := 0$. Let ST_j be the start time of activity j $(1 \leq j \leq n)$.

The nodes $1, \ldots, n$ of a directed graph are assigned to the activities $1, \ldots, n$ (and the activities are identified with the nodes). If there is a minimal time lag $T_{ij}^{\min} \geq 0$ between the start of two activities i and j, that is,

$$ST_j - ST_i \geq T_{ij}^{\min},$$

we introduce an arc $\langle i,j \rangle$ with weight $b_{ij} := T_{ij}^{\min}$. If there is a maximal time lag $T_{ij}^{\max}$ between the start of two activities i and j, that is,

$$ST_j - ST_i \leq T_{ij}^{\max}$$

we introduce an arc $\langle j,i \rangle$ with weight $b_{ji} := -T_{ij}^{\max}$ (see Fig. 1). In case of a maximal time lag $T_{ij}^{\max}$, we assume that there is a sequence of activities $i := j_0, j_1, \ldots, j_m := j$ and minimal time lags $T_{j_0 j_1}^{\min}, \ldots, T_{j_{m-1} j_m}^{\min}$ such that

$$T_{j_0 j_1}^{\min} + \ldots + T_{j_{m-1} j_m}^{\min} \leq T_{ij}^{\max}$$

In this way, we obtain a weakly connected weighted directed graph containing cycles, which we call an *MPM network.* Each arc with negative weight belongs to a cycle, and the length of any cycle is nonpositive. It is assumed

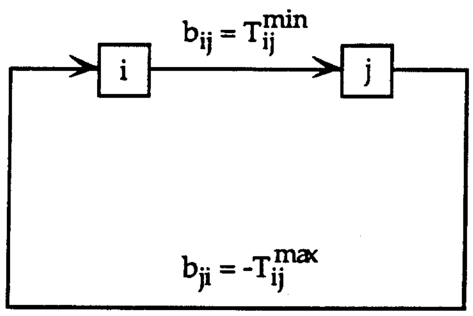

Fig. 1. Arcs for minimal and maximal time lags.

without loss of generality that each cycle contains at least one arc with positive weight. The node set of the MPM network N in question is denoted by V and the arc set by E where $V := \{1, \ldots, n\}$. Both the minimal and maximal time lags can be written in the form

$$ST_j - ST_i \geq b_{ij} \qquad (\langle i,j \rangle \in E).$$

Assume that the project begins at time zero, that is, $ST_1 := 0$. Then the project duration equals

$$ST_n = \max_{i=1,\ldots,n} (ST_i + D_i)$$

We note that prescribed *release dates* (also called *ready times*) and *deadlines* for activities can be expressed in terms of special minimal and maximal time lags. Let $r_j \geq 0$ be a given release date of activity j, that is, activity j is available for performing at time r_j. Then

$$ST_j - ST_1 \geq r_j$$

In other words, there is a minimal time lag $T_{1j}^{\min} = r_j$ between (the start of) activities 1 and j. Let $d_j \geq D_j$ be a given deadline for activity j, that is, activity j has to be completed by time d_j. Then

$$ST_j + D_j - ST_1 \leq d_j,$$

that is, there is a maximal time lag $T_{1j}^{\max} = d_j - D_j$ between (the start of) activities 1 and j.

In what follows, some additional concepts are needed. A strong component of an MPM network N (that is, a maximal subgraph where all nodes are reachable from each other) which contains at least one arc is called a *cycle structure.* A cycle structure represents, figuratively speaking, a maximal set of nested cycles.

If node j is reachable from node $i \neq j$, that is, there is a path from i to j, then i is termed a *predecessor* of j and j is a *successor* of i. Let i be a predecessor of j. The length of a longest path from i to j is denoted by l_{ij}. If there is an arc $\langle$i,j$\rangle$, then i is an *immediate predecessor* of j and j is an *immediate successor* of i.

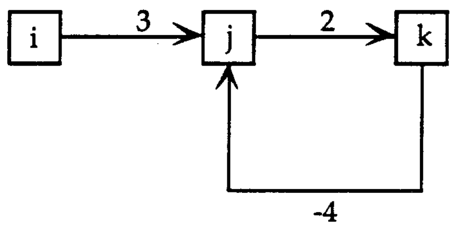

Fig. 2. Real and fictitious predecessors of nodes.

Node i is called a *real predecessor* of j either if $l_{ij} > 0$ or if $l_{ij} := 0$ and every arc $\langle\alpha,\beta\rangle$on a longest path from i to j has weight $b_{\alpha\beta} = 0$. Node i is called a *fictitious predecessor* of j either if $l_{ij} < 0$ or $l_{ij} := 0$ and there is at least one arc $\langle\alpha,\beta\rangle$ with weight $b_{\alpha\beta} < 0$ on every longest path from i to j. Analogously, real and fictitious successors as well as real and fictitious immediate predecessors and successors are defined. In Fig. 2, i is a real predecessor of both j and k, and k is a fictitious immediate predecessor of j. A real immediate predecessor i of an activity j has to be scheduled before j but not necessarily a fictitious immediate predecessor.

A node i of a cycle structure C is called an *initial node* of C if i does not have any real predecessor but has at least one real immediate successor in C. The first activity of a cycle structure C to be scheduled corresponds to an initial node of C. In Fig. 3, nodes 1 and 2 are initial nodes of the cycle structure.

3. Resource-constrained scheduling

Assume that the renewable resources $1, \ldots, K$ are required for carrying out the project in question. Let $R_\kappa > 0$ be the capacity of resource κ available and let $r_{i\kappa} \geq 0$ be the capacity of resource κ required for activity i ($\kappa := 1, \ldots, K$; $i \in V$), where R_κ and $r_{i\kappa}$ are supposed to be constant (independent of time). Of course, $r_{1\kappa} = r_{n\kappa} = 0$ for $\kappa = 1, \ldots, K$. It is assumed that $r_{i\kappa} \leq R_\kappa$ for $i := 2, \ldots, n-1$ and $\kappa = 1, \ldots, K$. Let

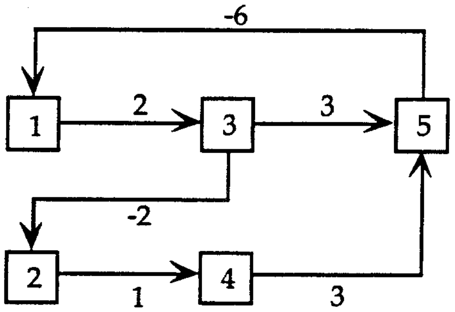

Fig. 3. Initial nodes of a cycle structure.

$$V(t) := \{i \in V | t - D_i < ST_i \leq t\}$$

be the set of activities in execution at time t or in time interval [t,t + 1] respectively. Assuming that the activities cannot be interrupted (nonpre-emptive case), the *resource-constrained project-scheduling problem* for the MPM network N in question is then as follows:

$$(N)\begin{cases} \min st_n \\ \text{such that } ST_j\text{-}ST_i \geq b_{ij} \quad (\langle i,j\rangle \in E) \\ \quad \sum_{i \in V(t)} r_{i\kappa} \leq R_\kappa \quad (\kappa = 1, \ldots, K;\ t = 0, 1, \ldots, T\text{-}1) \\ \quad ST_1 = 0 \\ \quad ST_i \in \mathbf{Z}_+ \ (i \in V) \\ \quad \text{activity splitting is not allowed} \end{cases}$$

where

$$T := \sum_{i \in V} \max(D_i, \max_{(i,j) \in E} b_{ij})$$

is an upper bound on the project duration.

To formulate the above optimization problem in a different way, binary variables x_{it} can be introduced where

$$x_{it} := \begin{cases} 1, \text{ if activity } i \text{ is in execution in the time interval} \\ (t-1, t) \\ 0, \text{ otherwise.} \end{cases}$$

Then problem (N) can be rewritten as a linear optimization problem with binary variables x_{it} ($i := 1, \ldots, m$; $t := 1, \ldots, D$) and integer variable D representing the project duration to be minimized.

Next, the resource-constraint project-scheduling problem is formulated for a cycle structure C of N with node set V_C and arc set E_C, where the 'partial project' C is treated as a separate entity and is started at time 0:

$$(C)\begin{cases} \min \max_{i \in V_C} (ST_i + D_i) \\ \text{such that} \qquad ST_j\text{-}ST_i \geq b_{ij} \quad (\langle i,j\rangle \in E_C) \\ \quad \sum_{i \in V_C(t)} r_{i\kappa} \leq R_\kappa \quad (\kappa = 1, \ldots, K; t = 0, 1, \ldots, T_C\text{-}1 \\ \quad ST_i \in \mathbf{Z}_+ \quad (i \in V_C) \\ \quad \text{activity splitting is not allowed} \end{cases}$$

where

$$V_C(t) := \{i \in V_C | t\text{-}D_i < ST_i \leq t\}$$

$$T_C := \sum_{i \in V_c} \max (D_i, \max_{\langle i,j\rangle \in E_c} b_{ij})$$

The following theorems can be proved (see also Bartusch *et al.*, 1988):

Theorem 1: the following problem is NP-hard: is there a feasible solution to (*C*) or respectively (*N*) and if yes, find a feasible solution.

It can be shown that, for a specific cycle structure *C*, the decision problem corresponding to (*C*) is equivalent to the partition problem and thus NP-complete. Recall that for acyclic MPM networks and for CPM networks, a feasible solution to (*N*) can be found in polynomial time.

Theorem 2: there is a feasible solution to (*N*) exactly if, for each cycle structure *C* of *N*, there is a feasible solution to (*C*).

Sketch of proof.

Necessity: Trivial.

Sufficiency: Later it will be shown how to find a feasible solution to (*N*) given a feasible solution to (*C*) for every cycle structure *C* in a relatively easy way. Hence, the computation of a feasible solution to (*C*) is the centre of our attention and, as we will see, causes some difficulties.

4. Algorithm AN for solving (N)

A heuristic procedure is presented which computes a feasible solution to (*N*).

Step 1. (Temporal analysis for *N*). Find earliest and latest start times EST_i and LST_i for all $i \in V$ without resource constraints;

Step 2. Find all cycle structures of *N*;

Step 3. (Temporal analysis for each cycle structure *C*). For each cycle structure *C*, compute earliest and latest start times EST_i^- and LST_i^- $(i \in V_c)$ without resource constraints;

Step 4 (Resource-constrained scheduling for each cycle structure *C*). For each cycle structure *C*

(1) Find a feasible solution $\{EST_i^+ | i \in V_c\}$ to (*C*) with earliest possible times EST_i^+ (that is, $\max_{i \in V_c}(EST_i^+ + D_i)$ as small as possible) using a heuristic. If no feasible solution is found, go to (2);

(2) Find a feasible solution $\{EST_i^+ | i \in V_c\}$ to (*C*) by an exact method (for example, by Bartusch's algorithm). If no feasible solution is found, stop.

Step 5 (Resource-constrained scheduling for network *N*). Compute a feasible solution $\{EST_i^+ | i \in V\}$ to (*N*) using a heuristic.

Concerning Step 4(2), it is noticed that in practice, a cycle structure *C* extremely rarely contains more than about 20 nodes. Thus, the computation of a feasible solution to (*C*) using an exact method does not cause too much computational effort.

5. Detailed discussion of the individual steps of algorithm AN

5.1. *Step 1*

It is well-known that EST_i is the length of a longest path from node 1 to node *i*. Moreover, if $LST_n := EST_n$, then $LST_n - LST_i$ is the length of a longest path from node *i* to node *n*. Longest paths can be found, for example, by using a label-correcting method that can be implemented in time $O(|V||E|)$ (cf. Gallo and Pallottino, 1986).

The times EST_i and LST_i are not needed for the following resource-constraint scheduling but only for static priority rules to sort the activities and, if schedulable, perform them observing the resource capacity available.

5.2. *Step 2*

To find the cycle structures, a well-known algorithm for determining the strong components in a digraph with time complexity $O(|E|)$ may be employed, which uses a recursive depth-first search procedure (cf. Even, 1979, Section 3.4).

5.3. *Step 3*

5.3.1. *Computation of the earliest start times in a cycle structure* C

For this we need a node that represents the beginning of the partial project corresponding to *C*. This auxiliary node designated by 0 and, for all $i \in V_C$, the arcs $\langle 0,i \rangle$ with weights $b_{0i} := 0$ are added to *C*. EST_i^- is equal to the length of a longest path from 0 to *i* in the 'extended' cycle structure *C*.

Note: given the times EST_i^-, the initial nodes of *C* (required for the following resource-constrained scheduling) can be found with relative ease. It can be shown that a node *s* in *C* without a real immediate predecessor and with at least one real immediate successor in *C* represents an initial node of *C* exactly if $EST_s^- = 0$ (that is, the determination of longest paths, which are needed for finding non-immediate predecessors, is not necessary).

5.3.2. *Computation of latest start times* LST_i^- *in* C

Let *s* be an initial node of *C* and LST_i^s be the latest start time of activity *i* in *C* subject to $LST_s^s := 0$. Note that for an activity *i* and different initial nodes *s*, the times LST_i^s may be distinct. LST_i^s can be computed as follows: let l_{is} be again the length of a longest path from *i* to *s* in *C*. Since $-l_{is} \geq 0$ is the maximal time lag between (the beginning of) activities *s* and *i*, it holds that $LST_i^s = -l_{is}$ (compare Fig. 4).

Let

$$d_s := \max_{i \in V_c} (LST_i^s + D_i)$$

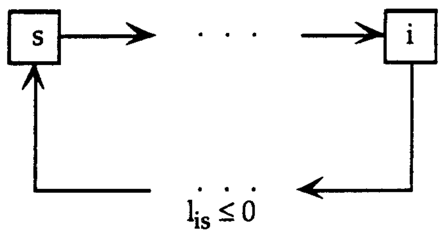

Fig. 4. Maximal time lag between activities s and i.

be the duration of partial project C starting with initial node or respectively initial activity s at time 0. Let s^* be an initial node s with maximum d_s; s^* is called an *optimal initial node* of C, and we set

$$LST_i^- := LST_i^{s^*} \ (i \in V_c)$$

The first activity to be scheduled within C corresponds to an optimal initial node s^*. The time complexity of step 3 is $O(|V_C||E_C|)$.

5.4. *Step 4 (Resource-constrained scheduling for cycle structure* C*)*

5.4.1. *Priority rules for projects without maximal time lags*

The basic idea of a priority-rule method for acyclic MPM networks is the following: at the current point in time t, the set M of eligible activities is figured out. An activity j is called *eligible* at time t if:

(1) j has not been scheduled yet by time t;
(2) $ST_i + b_{ij} \leq t$ for all immediate predecessors i of j.

By definition, the activities that have been in execution at time $t-1$ but have not been completed yet and one thus has to go on performing them at time t (recall that activities must not be interrupted) are not called eligible.

At first the latter activities are scheduled at time t. To decide upon which additional activities from set M are to be scheduled at time t (that is, their execution is to begin at time t), we introduce a *total order* $\preccurlyeq$ (a binary, reflexive, transitive and antisymmetric relation) on M. The activities from M are then scheduled at time t in that order as long as resource capacity is available.

Such a total order can be defined as follows: we assign a *sorting characteristic* or *priority* G_j to each activity j. This sorting characteristic may depend on

(1) The structure of the network (for example, the number of successors of node j, $|S(j)|$);
(2) The temporal analysis (for example, the total float of activity j, $LST_j - EST_j$);
(3) The activity duration D_j;
(4) The resource requirements $\Sigma_{\kappa=1}^{K} r_{j\kappa}$ for activity j.

More than 20 sorting characteristics have been examined in Zhan (1991).

Then $i \preccurlyeq j$ ('i is scheduled before j') exactly if

(1) $G_i < G_j$ or
(2) $G_i = G_j$ and $i < j$.

For some sorting characteristics (for example, the resource requirements $\Sigma_{\kappa=1}^{K} r_{j\kappa}$), ' $<$ ' has to be replaced by ' $>$ ' in (1). Moreover, instead of the activity number j, a different attribute may be used as secondary sorting characteristic in (2).

5.4.2. *Priority rules for projects with maximal time lags*

In cyclic MPM networks, an unscheduled activity j is not necessarily eligible at time t if $ST_i + b_{ij} \leq t$ for all real immediate predecessors i of j due to maximal time lags. Let us look at two examples.

Example 1.

Suppose activity j belongs to a cycle structure C and does not have any real immediate predecessor in C, but has a real non-immediate predecessor i in C, that is, $l_{ij} > 0$ (see Fig. 5). Within C, activity j cannot be scheduled at time 0 but at time

$$MINST_j := l_{ij}$$

at the earliest.

Example 2.

If activity j has the fictitious immediate successor i, then j has to be scheduled at time

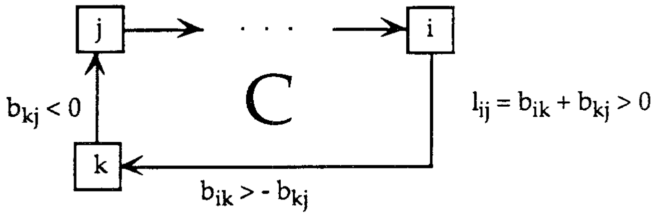

Fig. 5. Example of positive lower bound $MINST_j$.

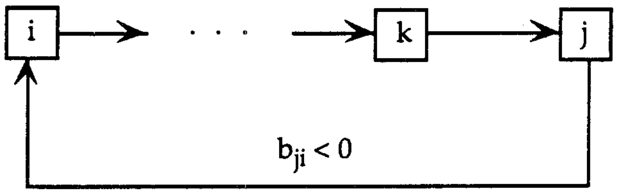

Fig. 6. Example of upper bound $MAXST_j$.

$$MAXST_j := ST_i + |b_{ji}|$$

at the latest (maximal time lag!), cf. Fig. 6. In other words, activity j cannot be scheduled at time t if $t > MAXST_j$. To indicate that the latest start of activity j at time $MAXST_j$ is caused by activity i, we write $MAX_j := i$.

Note: in analogy, each real immediate predecessor k of j has to be scheduled at time $MAXST_k := MAXST_j - l_{kj}$ at the latest, and we put $MAX_k := i$.

The exact specification and updating of the lower and upper bounds $MINST_j$ and $MAXST_j$, respectively, on the start time of an activity j will be done in the course of the heuristic procedure.

Taking the above considerations into account, an activity j is called *eligible* at time t if

(1) j has not been scheduled yet by time t;
(2) $ST_i + b_{ij} \leq t$ for all real immediate predecessors i of j;
(3) $MINST_j \leq t \leq MAXST_j$.

The set of activities eligible at time t is again denoted by M.

Now a total order on M is defined which, in addition to the above sorting characteristic G, takes into account the upper bound $MAXST$ if meeting that bound is at risk due to limited resources. This total order is denoted by $\preccurlyeq|MAXST$. The total order for acyclic MPM networks introduced above will be denoted by $\preccurlyeq|G$ in what follows.

For $j \in V$, let $\Delta_j \in \mathbf{Z}_+$ such that $0 \leq \Delta_j \leq \max_{i \in V} i$. Then $i \preccurlyeq j|MAXST$ ('i is scheduled before j') exactly if one of the four following conditions is satisfied:

(1) $t + \Delta_i > MAXST_j$, $t+\Delta_j > MAXST_i$, and $MAXST_i < MAXST_j$;
(2) $t+\Delta_i > MAXST_j$, $t+\Delta_j > MAXST_i$, $MAXST_i = MAXST_j$, and $i \preccurlyeq j|G$;
(3) $t+\Delta_i \leq MAXST_j$ and $t+\Delta_j > MAXST_i$;
(4) $t+\Delta_i \leq MAXST_j$, $t+\Delta_j \leq MAXST_i$, and $i \preccurlyeq j|G$.

$t+\Delta_i > MAXST_j$ says that scheduling activity i at time t possibly requires resources for activity i beyond time $MAXST_j$. Thus, the upper bound $MAXST_j$ for activity j is at risk. Hence, the meaning of conditions (1) to (4) is as follows:

(1) and (2) The upper bounds for both activities i and j are at risk, each bound by scheduling the other activity;
(3) Only one upper bound is at risk by scheduling the other activity;
(4) None of the two upper bounds is at risk.

It can be shown that $\preccurlyeq|MAXST$ really represents a total order if the Δ_j's are equal for all $j \in M$. Thus, it is recommended to choose

$$\Delta_j = \Delta \text{ for all } j \in V \text{ where } 0 \leq \Delta \leq \max_{i \in V} D_i$$

For smaller Δ, the order in which the activities from M are scheduled more often depends only on the sorting characteristic G. If we choose $\Delta_j = D_j$ for every $j \in V$, the relation $\preccurlyeq|MAXST$ is not necessarily transitive.

5.4.3. *Backward scheduling*

Scheduling the activities from M according to the total order $\preccurlyeq|MAXST$ as long as resource capacity is available does not guarantee that each activity j will be scheduled by time $MAXST_j$.

If we find out at time t that $t > MAXST_j$ for some unscheduled activity j and if $MAX_j = i$ (that is, the upper bound $MAXST_j$ is due to activity i), scheduling of all activities between times ST_i and t is nullified and performed again. To assign a larger value to $MAXST_j$ as required, we choose $MINST_i > ST_i$, in particular

$$ST_i + 1 \leq MINST_i \leq ST_i + (t - MAXST_j)$$

Note: often $MINST_i := ST_i + 1$ is chosen hoping that this suffices to schedule the real predecessors of j sooner due to available resources and to meet the deadline $MAXST_j$.

5.4.4. *Algorithm AC for step 4(1)*

Let L be the set of scheduled activities from cycle structure C. Then step 4(1) of algorithm AN corresponds to the following.

(Determine $\{EST_j^+ | j \in V_c\}$, i.e. earliest start times in C with resource constraints)

Input.

EST_j^- and LST_j^- for all $j \in V_c$
$\hat{T} := \max_{i \in V_c} LST_i^-$

Step i (Initialization). Set $t := 0$, $L := \{s\}$ (s optimal initial node), $EST_s^+ := 0$, and $EST_j^+ := -1$, $MINST_j := EST_j^-$, $MAXST_j := LST_j^-$, $MAX_j := 0$ for all $j \in V_C \backslash \{s\}$ ($A \backslash B := \{a \in A | a \notin B\}$ is the difference of sets A and B);

Step ii (Eligible set M). If $L = V_C$, stop (all activities have been scheduled). If there is some $j \in V_C \backslash L$ with $MAXST_j < t$, go to step vi (backward scheduling). Construct set M of activities eligible at time t. If $M = \emptyset$, go to step v (determine new time t);

Step iii (Ordering of M). Order M according to $\preccurlyeq|MAXST$;

Step iv (Construction of schedule).

(a) Take the next $j \in M$. If, for some resource κ, $r_{j\kappa}$ exceeds the remaining capacity, go to step v

(b) Set $L := L \cup \{j\}$, $EST_j^+ := t$. Update $MAXST_i$ and MAX_i for all $i \in V_C \backslash L$ (i.e. if $i \in V_C \backslash L$ is a fictitious predecessor of j with $MAXST_i > t - l_{ij}$, then set $MAXST_i := t - l_{ij}$ and $MAX_i := j$). Go to (a)

Table 1. Earliest and latest times for the example of Fig. 7

i	EST_i^-	LST_i^-	$MINST_i$	$MAXST_i$	EST_i^+
2	0	2	0	2	0
3	2	4	2	4	3
4	2	6	2	6, 3	2
5	6	8	6	8, 7	7

Step v (Determine a new time t). Let τ be the minimum of all $EST_i^+ + D_i > t$ and $EST_i^+ + b_{ij} > t$ with $i \in L$ and $\langle i,j \rangle \in E_C$. Set $t := \tau$. If $t > \hat{T}$, stop (there is no feasible solution); otherwise go to step ii;

Step vi (Backward scheduling). Set i := MAX_j. If $i = 0$, stop (there is no feasible solution); otherwise reconstruct the state at time EST_i^+ (i.e. for all $k \in L$ with $EST_k^+ \geq EST_i^+$, set $L := L \setminus \{k\}$, $MINST_k := EST_k^-$, $MAXST_k := LST_k^-$, $MAX_k := 0$). Set $MINST_i := EST_i^+ + \theta$ with $1 \leq \theta \leq t - MAXST_j$. Set $t := EST_i^+$ and go to step ii.

Algorithm AC without backward scheduling can be implemented to run in $O(|E_C||T||\log|E_C|)$ time.

5.4.5. *Example*

To illustrate algorithm AC the cycle structure shown in Fig. 7 is considered, where there is a single resource of capacity 5 available and r_i is the amount of resource required by activity i.

Node 2 is the only initial node of the cycle structure. By step 3 of algorithm AN, we obtain the quantities EST_i^- and LST_i^- in Table 1. The initial values of $MAXST_i$, which are equal to LST_i^-, are given before the comma in column 5 of Table 1. The values of $MAXST_i$ after the comma and the quantities EST_i^+ are computed by algorithm AC. $\hat{T} = \max_{i \in V_C} LST_i^- = 81$ is an upper bound on the start time of any activity.

For the order $\preccurlyeq |MAXST$, we choose $\Delta \nabla 3$. The first scheduling time is $t = 0$.

$t = 0$. Activity 2 is scheduled, that is, $EST_2^+ := 0$. Moreover, we set $MAXST_4 := 3$ and $MAX_4 := 2$ (compare step iv of algorithm AC). The next scheduling time is $t = 2$ (cf. step v of algorithm AC);

$t = 2$. The set of eligible activities is $M = \{3,4\}$. Since $t + \Delta \;\; \nabla\, 5 > MAXST_3 = 4$, $t + \Delta > MAXST_4 = 3$, and $MAXST_4 < MAXST_3$, activity 4 is scheduled before activity 3 (see condition (1) of order $\preccurlyeq |MAXST$). Because of $D_2 = 3$, activity 2 is still in execution at time 2. Thus, the remaining capacity is 3 and from activities 3 and 4, only activity 4 can be scheduled at time 2: $EST_4^+ := 2$. The next scheduling time is $t = 3$;

$t = 3$. Both activities 2 and 4 are completed at time 3. Thus, the eligible activity 3 can be scheduled at time 3: $EST_3^+ := 3$. Moreover, we set $MAXST_5 := 7$ and $MAX_5 := 3$. The next scheduling time is $t = 4$;

$t = 4$. There is no eligible activity because the remaining activity 5 can be scheduled at time $EST_3^+ + T_{35}^{\min} = 7$ at the earliest;

$t = 7$: Activity 5 is scheduled: $EST_5^+ = 7$.

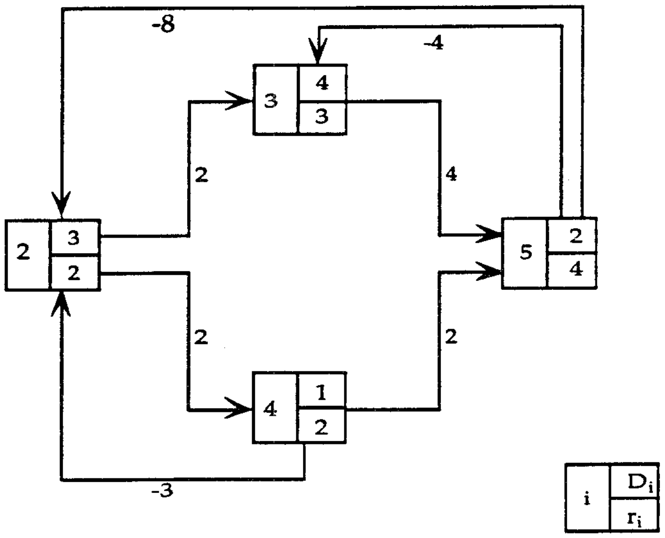

Fig. 7. Example of a cycle structure.

5.5. *Step 5 (Resource-constrained scheduling for network* N)

To determine a feasible solution for the whole MPM network N, each cycle structure in N is replaced by a single node or respectively activity and a feasible solution is computed for the resulting acyclic MPM network N'. This means that all activities of a cycle structure are scheduled jointly. If we have found a feasible solution for each cycle structure of N, we also obtain a feasible solution for the whole network N. In detail, we proceed as follows: let $C_1, \ldots, C_r$ be the cycle structures of N, and let $\{EST_i^+ \,|\, i \in V_{C_\rho}\}$ be the feasible solution to the resource-constrained scheduling problem for cycle structure C_ρ ($\rho = 1, \ldots, r$). Moreover, let

$$V^C := \bigcup_{\rho=1}^{r} V_{C_\rho} \text{ and } E^C := \bigcup_{\rho=1}^{r} E_{C_\rho}$$

be the sets of all nodes and arcs, respectively, of N that belong to some cycle. Each cycle structure C_ρ is replaced by a node or respectively activity c_ρ with duration

$$D_{c_\rho} := \max_{i \in V_{C_\rho}} (EST_i^+ + D1_i)$$

and resource requirements

$$r_{c_\rho \kappa} := \max_{0 \leq t \leq D_{C_\rho}} \sum_{i \in V_{C_\rho}(t)} r_{i\kappa} \qquad (\kappa = 1, \ldots, K)$$

where

$$V_{C_\rho}(t) := \{i \in V_{C_\rho} | t - D_i < EST_i^+ \leq t\}$$

Network N is then replaced by the acyclic network N' with node set

$$V' := (V \backslash V^C) \cup \{c_1, \ldots, c_r\}$$

and arc set E' which results from set $E \backslash E^C$ by the following arc replacement (compare Fig. 8):

$i \in V_{C_\rho}, j \in V \backslash V^C$: *replace* (i,j) by $\langle c_\rho, j \rangle$
with weight $b_{C_\rho j} := b_{ij} + D_{c_\rho}$
$i \in V_{C_\rho}, j \in V_{C_\sigma}$: replace (i,j) by $\langle c_\rho, c_\sigma \rangle$
with weight $b_{c_\rho c_\sigma} := b_{ij} + D_{c_\rho}$
$i \in V \backslash V^c, j \in V_{C_\sigma}$: replace (i,j) by $\langle i, c_\sigma \rangle$
with weight $b_{ic_\sigma} := b_{ij}$.

If parallel arcs result, the one with the largest weight is selected.

A feasible solution $\{ST'_i | \in V'\}$ for N' can then be computed using a priority-rule method for acyclic MPM networks (or for CPM networks) or a simplified version of algorithm AC (without the quantities $MINST_i$, $MAXST_i$, MAX_i related to cycles). Finally, a feasible solution $\{ST_i | i \in V\}$ for the original network N is obtained as follows:

$$ST_i := \begin{cases} ST_i' \text{ if } i \in V \backslash V^C \\ ST'_{c_\rho} + EST_i^+ \text{ if } i \in V_{C_\rho} \ (\rho = 1, \ldots, r) \end{cases}$$

5.6. *Supplement*

Several modifications of steps 4 (problem (C) and 5 (problem (N) have been investigated in Zhan (1991). For example, we can try to postpone the earliest start times EST_i^+ for $i \in V_C$ as far as possible (that is, we determine corresponding latest start times LST_i^+) and thus reduce the resource requirements per unit time of C. This may permit us to schedule several cycle structures jointly and possibly results in a 'better' feasible solution to (N).

Instead of scheduling all activities of a cycle structure jointly, we may try to schedule these activities one after another. This may result in better exploitation of the resource capacities available.

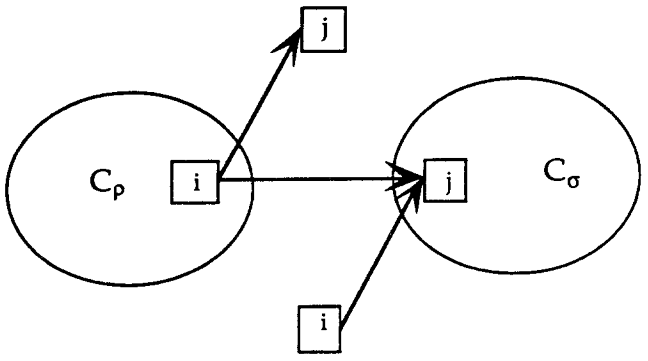

Fig. 8. Reduction of cycle structures.

6. Numerical results

Zhan (1991) has performed computational experiments to test the performance of the heuristic procedure AN with 26 different sorting characteristics. The MPM networks have been generated randomly by a network generator. A Hewlett Packard workstation of type HP 9000/360 was used. Three classes of networks have been investigated:

(1) Small networks containing up to 50 activities and requiring one resource;

(2) Large networks with 100–1000 activities and one resource;

(3) Small and medium-size networks containing up to 300 activities and requiring three resources.

For a fixed number of activities, 80 different networks have been studied. To find representative samples, the networks of given size and number of resources have been classified according to some network measures such as ratio of arcs to nodes, number of maximal time lags, number and size of cycle structures, and ratio of activity durations to minimal and maximal time lags (for details we refer to Zhan, 1991).

6.1. *Results for class 1*

The priority-rule methods have been compared with Bartusch's exact algorithm. Often, Bartusch's method did not come up with an optimal solution within a running time of 2 hours, in particular, for larger networks. The best heuristic procedures provided feasible solutions within about 2% of optimality (referring to the objective function value) requiring a few seconds of computation time.

6.2. *Results for classes 2 and 3*

Different heuristics using different priority rules (or respectively sorting characteristics) have been compared (by means of the Wilcoxon signed rank test, cf. Lawler *et al.*, 1985, Section 7.2). The following results were obtained:

(1) In general, good priority rules (as far as the quality of the approximate solution is concerned) for CPM networks (that is, projects without maximal time lags) represent good priority rules for cyclic MPM networks (that is, networks with maximal time lags), too. Good sorting characteristics are, for example:

Table 2. Computational results

	Average computing time (s)				*Average deviation (%)*			
n	*MTS*	*LFT*	*BCH*[*]	*LTS*	*MTS*	*LFT*	*BCH*	*LTS*
50	0.62	0.88	3.05	0.48	2.10	1.57	1.15	5.06
100	2.71	3.11	12.0	1.83	2.10	1.00	0.33	7.95
300	3.45	18.5	60.0	2.21	1.81	0.73	0.05	13.9
1000	16.8	26.0	102	10.4	1.19	0.45	0.07	14.4

[*]BCH = best combined heuristic

MTS ('most total successors' first: activities with largest number of successors have highest priority);

LST ('latest start time' first: activities with smallest latest start time have highest priority);

LFT ('latest finish time' first: activities with smallest latest finish time have highest priority);

LFS ('least float per successor' first: activity j with smallest value of $(1/|IS(j)|)\sum_{i\in IS(j)}(LST_i - EST_i)$ has highest priority, where $IS(j)$ is the set of the immediate successors of j);

(2) Priority rules 'opposite' to good rules are generally poor, for example:

LTS ('least total successors' first);

EST ('earliest start time' first);

EFT ('earliest finish time' first).

The rules

SPT ('shortest processing time' first)

LPT ('longest processing time' first),

which are often used in practice, have also turned out to be poor;

(3) 'Combined' heuristics (using four different heuristics and selecting the best of the feasible solutions obtained) are much better than the individual heuristics. In particular, combined heuristics behave much better than single heuristics for specific 'pathological' networks, that is, there are much fewer 'runaways';

(4) As the number of activities increases, good heuristics provide better solutions (that is, solutions with project durations closer to the shortest project duration found by any heuristic) and poor heuristics provide worse solutions;

(5) The computing time generally increases linearly with the number of activities, where the factor of proportionality is a little less than one. For networks with 1000 activities and one resource the average computing time (of a single heuristic) is less than 30 s;

(6) For three resources, the average computing time is almost twice as large as for one resource if every activity requires two resources on the average.

Table 2 shows some computational results for one resource, where for each number n of activities, 80 problems have been solved (cf. Zhan, 1991). The deviation refers to the difference between the project duration for the solution in question and the shortest project duration found by any heuristic. Note that the rules MTS and LFT provide good heuristics whereas LTS yields a poor one. For three resources, the computational results are quite similar except that, as already mentioned, the average computing time is about twice as large as for one resource.

References

Alvarez-Valdés, R. and Tamarit, J. M. (1989) Heurisitic algorithms for resource-constrained project scheduling: a review and an empirical analysis, in *Advances in Project Scheduling*, Slowinski, R. and Weglarz, J. (eds), Elsevier, Amsterdam, pp. 113–134.

Bartusch, M. (1983) Optimierung von Netzplänen mit Anordnungsbeziehungen bei knappen Betriebsmitteln, Ph. D. Thesis, RWTH Aachen, Germany.

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

Christofides, N., Alvarez-Valdés, R. and Tamarit, J. M. (1987) Project scheduling with resource constraints: a branch and bound approach. *European Journal of Operational Research*, **29,** 262–273.

Davis, E. W. (1975) Project network summary measures constrained-resource scheduling. *AIIE Transactions*, **7,** 132–142.

Davis, E. W. and Patterson, J. H. (1975) A comparison of heuristic and optimum solutions in resource-constrained project scheduling. *Management Science*, **21,** 944–955.

Demeulemeester, E. and Herroelen, W. (1992) A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management Science*, **38,** 1803–1818.

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

Elsayed, E. A. (1982) Algorithms for project scheduling with resource constraints. *International Journal of Production Research*, **20,** 95–103.

Even, S. (1979) *Graph Algorithms*, Pitman, London.

Gallo, G. and Pallottino, S. (1986) Shortest path methods: a unifying approach. *Mathematical Programming Study*, **26,** 38–64.

Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G. and Shmoys, D. B. (eds) (1985) *The Traveling Salesman Problem*, John Wiley, New York.

Neumann, K. and Morlock, M. (1993) *Operations Research*, Carl Hanser, München.

Patterson, J. H. (1984) A comparison of exact approaches for solving the multiple constrained resource project scheduling problem. *Management Science*, **30,** 854–867.

Patterson, J. H., Slowinski, R., Talbot, F. B. and Weglarz, J. (1989) An algorithm for a general class of precedence and resource constrained scheduling problems, in *Advances in Project Scheduling*, Slowinski, R. and Weglarz, J. (eds), Elsevier, Amsterdam, pp. 3–28.

Roy, B. (1964) *Les problèmes d'ordonnancement*, Dunod, Paris.

Stinson, J. P., Davis, E. W. and Khumawala, B. M. (1978) Multiple resource-constrained scheduling using branch and bound. *AIIE Transactions*, **10,** 252–259.

Talbot, F. B. and Patterson, J. H. (1978) An efficient integer programming algorithm with network cuts for solving resource-constrained scheduling problems. *Management Science*, **24,** 1163–1174.

Zhan, J. (1991), Heuristische Ressourcenplanung in MPM-Netzplänen mit beschränkter Kapazität, Ph. D. Thesis, University of Karlsruhe, Germany.