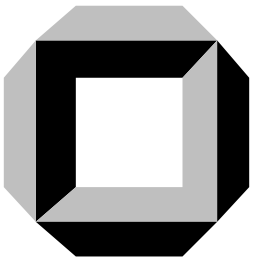
INSTITUT FÜR WIRTSCHAFTSTHEORIE
UND OPERATIONS RESEARCH
UNIVERSITÄT KARLSRUHE

Generation of Resource-Constrained Project Scheduling Problems with Minimal and Maximal Time Lags

Christoph Schwindt

Report WIOR-489



TECHNICAL REPORT

Kaiserstraße 12 · D - 76128 Karlsruhe · Germany

Generation of Resource-Constrained Project Scheduling Problems with Minimal and Maximal Time Lags

Christoph Schwindt

Report WIOR-489

November 1996

This research was done within the Research Group Resource–Constrained Project Scheduling (http://www.wior.uni-karlsruhe.de/RCPSP/), which is partially supported by the Deutsche Forschungsgemeinschaft (DFG).

Abstract

We describe a new problem generator for three different types of resource-constrained project scheduling problems with minimal and maximal time lags: the project duration problem, the resource leveling problem, and the resource investment problem. The cyclicity of the underlying project network requires special techniques for the parameter-driven generation of cycle structures. Two different approaches to the generation of cycle structures are presented: the direct method which adds backward arcs to an acyclic network and the contraction method which constructs isolated strong components which are then contracted to a single node and incorporated into an acyclic network. Efficient algorithms for the direct method and the contraction method are provided.
The generation of the project's basic data as well as the resource demand and availability is based on ProGen by Kolisch et al.
A computational experiment relating computation times for the solution of the project duration problem to several control parameters shows the strong impact of parameter configuration on problem hardness.

Key Words

Problem generator, activity-on-node networks, network measures, resource-constrained project scheduling, maximal time lags.

Contents

1. Introduction

Several network and problem generators for resource-constrained scheduling problems are known from literature (Kolisch et al. 1995, Demeulemeester et al. 1993, Agrawal et al. 1994). One of the best-known scheduling problems is the Resource-Constrained Project Scheduling Problem (RCPSP). Until 1992, an inhomogeneous testset of Patterson (1984) has been used as a benchmark for algorithms. This testset includes problems published in Davis (1969), Patterson and Huber (1974), Davis and Patterson (1975), Talbot and Patterson (1978), and Patterson (1984). Of course, these problems were not generated using a unified and systematic approach controled by several network and resource-based parameters like network complexity or resource strength.

Moreover, the development of a new efficient branch-and-bound procedure for RCPSP by Demeulemeester and Herroelen (1992) has shown that all problems of the Patterson testset without exception belong to a class of "easy" problems, that is, they can be solved optimally within a very short amount of time. Kolisch et al. (1995) have shown that there are lower-sized problems which are much harder to solve.

Therefore, the empirical analysis of algorithms should be based on problem instances which have been generated systematically by a problem generator. The performance of the tested algorithms can then be evaluated depending on different problem measures.

The problem generator ProGen of Kolisch et al. (1995) creates problem instances of RCPSP or the generalized multi-mode problem MRCPSP. Several network measures such as the number of nodes, the network complexity, the number of predecessors and successors of a node as well as parameters for the generation of the basic data and the resource constraints can be specified.

The network generator of Demeulemeester et al. (1993) creates acyclic weakly connected digraphs where each network structure (with given number of nodes and arcs) can be generated with exactly the same probability (strongly randomized networks). Due to the specific approach required for the strong randomness, other network measures such as redundancy cannot be observed.

Whereas the networks generated by Kolisch et al. (1995) and Demeulemeester et al. (1993) are so-called activity-on-node networks (A-on-N networks, that is, the activities are identified with the nodes of the project network, the arcs define time and precedence constraints), the generator of Agrawal et al. (1994) constructs activity-on-arc networks (A-on-A networks) for which the arcs correspond to the activities of a project. Control parameters are the number of nodes, the number of arcs, and the *CI*-index of reduction complexity (which is defined to be the minimum number of node reductions sufficient to reduce a series-parallel digraph to a single edge, cf. Bein et al. 1992).

An important generalization of RCPSP is problem RCPSP/max where maximal time lags between the start of activities define additional time constraints. Maximal time lags can, for instance, be used to model due dates, time-varying resource demands of activities, or time windows due to technological or organizational restrictions. For applications we refer to Neumann and Schwindt (1995).

A maximal time lag $-T_{ij}^{max}$ between activities i and j can be represented by a backward arc $< j,i >$ from the node corresponding to activity j to the node corresponding to activity i in the underlying A-on-N project network $\dot{N} =< V,E;c >$. $< j,i >$ is weighted with . The introduction of backward arcs, however, generates cycle structures in $\dot{N}$. Thus, project networks modeling precedence and time constraints of RCPSP/max instances are no

longer acyclic and we need specific techniques for the parameter-driven generation of cycle structures.
The new problem generator ProGen/max developed within this paper is based on the methodology of ProGen for the basic data generation and the construction of acyclic network structures. The generation of resource demand and availability is adopted from ProGen without almost any modification and will only be sketched briefly. The main emphasis of this paper is on theoretical results for cyclic digraphs and methods incorporated in ProGen/max for an efficent construction of cyclic networks taking into account several measures like number, size, and density of cycle structures.

The remainder of this paper is organized as follows. Section 2 is concerned with three different optimization models for resource-constrained project scheduling with minimal and maximal time lags, basic definitions of graph theory, theorems which will be used for the network generation algorithms, and several network measures which are known from literature. Two different approaches to the construction of cyclic networks, the direct and the contraction method, are presented in Section 3. In Section 4 we summarize the generation of resource constraints developed by Kolisch et al. (1995). Section 5 deals with the evaluation of the relationship between problem parameters and the hardness of problem instances.

The problem generation can be outlined as follows:

Algorithm A1. Problem generation

(1) Basic data generation (cf. Subsection 3.1)

(2) Construction of the structure of the underlying project network (cf. Subsection 3.2)

(3) Determination of the activity durations (cf. Subsection 3.3)

(4) Determination of minimal and maximal time lags between activities (cf. Subsection 3.3)

(5) Generation of resource availability and resource demand (cf. Section 4)

2. Basic Concepts

2.1 Models for resource-constrained scheduling

ProGen/max generates instances of several types of multi-mode resource-constrained project scheduling problems with minimal and maximal time lags and renewable, non-renewable, and doubly-constrained resources: The project duration problem MRCPSP/max, the resource-leveling problem MRLP/max, and the resource investment problem MRIP/max. Of course, special cases, such as MRCPSP or RLP/max can be obtained as well, by fixing the number of maximal time lags to zero or, respectively, by restricting the number of possible execution modes for any activity to one.
In this subsection we give a formal definition of the three problems MRCPSP/max, MRLP/max , and MRIP/max. For more details and formulations as linear optimization problems with binary variables, we refer to Franck and Schwindt (1996).

We introduce the following notation:

0, n+1	dummy activities representing the start and the end of the project, respectively ($D_0 = D_{n+1} = 0$)
b_{jlm}	weight of arc $< j,l > \in E$ ($m \in M_j$) with $b_{jlm} := \begin{cases} T_{jlm}^{min}, & \text{if there is a minimal time lag between activities } j \text{ and } l \\ -T_{lj}^{max}, & \text{if there is a maximal time lag between activities } l \text{ and } j \end{cases}$
c_i	integer-valued cost of one unit of resource i ($i \in R^{\rho} \cup R^{\nu}$)
D_{jm}	non-preemptable integer-valued duration of activity j scheduled in mode m ($j \in V, m \in M_j$)
E	arc set of the underlying project network $\dot{G} =< V,E >$
M_j	set of modes in which activity j can be performed ($j \in V$)
R^{ρ}	set of renewable and doubly-constrained resources
R^{ν}	set of nonrenewable and doubly-constrained resources
R_i^{ρ}	integer-valued per period capacity of renewable (doubly-constrained) resource i ($i \in R^{\rho}$)
R_i^{ν}	integer-valued total capacity of nonrenewable (doubly-constrained) resource i ($i \in R^{\nu}$)
r_{ijm}^{ρ}	integer-valued per period usage of renewable (doubly-constrained) resource i performing activity j in mode m ($j \in V, i \in R^{\rho}, m \in M_j$)
r_{ijm}^{ν}	integer-valued total consumption of nonrenewable (doubly-constrained) resource i performing activity j in mode m ($j \in V, i \in R^{\nu}, m \in M_j$)
ST_j	start time of activity j ($j \in V$)
T	fixed maximum project duration
$\overline{T}$	upper bound on the project duration

$T_{jlm}^{min}, T_{jl}^{max}$	minimal and maximal integer-valued time lags, respectively, between the start of activities *j* and *l*. The minimal time lag is depending on execution mode *m* of activity *j*
$V = \{0,\ldots,n+1\}$	set of activities which are to be performed. *V* at the same time corresponds to the node set of the underlying project network $\dot{G} =< V,E >$
$V(t)$	set of activities which are in progress at time *t*
x_{jm}	binary variable which is exactly one, if activity *j* is performed in mode *m*, zero otherwise $(j \in V, m \in M_j)$

MRCPSP / max can be stated as follows:

$$
(2.1)\quad (\text{MRCPSP}/\max)\quad
\begin{cases}
\min & ST_{n+1} & \\
\text{s.t.} & ST_l - ST_j \geq \sum\limits_{m\in M_j} b_{jlm}x_{jm} & (< j,l >\in E) \\
& \sum\limits_{j\in V(t)}\sum\limits_{m\in M_j} r_{ijm}^{\rho}x_{jm} \leq R_i^{\rho} & (i \in R^{\rho}, t = 0,\ldots,\overline{T}-1) \\
& \sum\limits_{j\in V}\sum\limits_{m\in M_j} r_{ijm}^{\nu}x_{jm} \leq R_i^{\nu} & (i \in R^{\nu}) \\
& \sum\limits_{m\in M_j} x_{jm} = 1 & (j \in V) \\
& x_{jm} \in \{0,1\} & (j \in V, m \in M_j)
\end{cases}
$$

The project duration is to be minimized. Minimal and maximal time lags between activities as well as limited availabilities of renewable, nonrenewable, and doubly-constrained resources have to be taken into account. Any job is performed in exactly one mode.

The resource leveling problem MRLP / max consists of the determination of a feasible schedule which minimizes a monotonously increasing function of the variation in time of resource requirements. Different objective functions can be found in literature. An objective function with many applications in practice is, for instance,

$$
(2.2)\quad f(ST_1,\ldots,ST_n) := \sum_{i\in R^{\rho}} c_i \max_{t=0,\ldots,T-1} \sum_{j\in V(t)} r_{ijm}^{\rho}x_{jm}
$$

which corresponds to the mean resource availability which has to be provided if the resource availability (which is constant in time) is determined by the maximum resource requirements during the execution of the project.

MRLP / max can be stated as follows:

(2.3) (MRLP / max)

$$
\begin{cases}
\min & f(ST_1,\ldots,ST_n) & \\
\text{s.t.} & ST_l - ST_j \geq \sum_{m\in M_j} b_{jlm}x_{jm} & (<j,l>\in E)\\
& ST_{n+1} \leq T & \\
& \sum_{j\in V(t)}\sum_{m\in M_j} r^{\rho}_{ijm}x_{jm} \leq R^{\rho}_i & (i\in R^{\rho}, t=0,\ldots,\overline{T}-1)\\
& \sum_{j\in V}\sum_{m\in M_j} r^{\nu}_{ijm}x_{jm} \leq R^{\nu}_i & (i\in R^{\nu})\\
& \sum_{m\in M_j} x_{jm} = 1 & (j\in V)\\
& x_{jm}\in\{0,1\} & (j\in V, m\in M_j)
\end{cases}
$$

A leveling objective function f of schedule $(ST_1,\ldots,ST_n)$ is to be minimized. Analogously to MRCPSP / max, precedence, time, and resource constraints must be observed and each activity is performed in exactly one mode. Additionally, the project has to be completed by a given point in time T.

The resource investment problem MRIP / max can be viewed as some kind of dualization of MRCPSP / max (cf. Demeulemeester 1992). The objective is the minimization of the costs for resource availability subject to the punctual completion of the project.

MRIP / max can be stated as follows:

(2.4) (MRIP / max)

$$
\begin{cases}
\min & \sum_{i\in R^{\rho}} c_i R^{\rho}_i + \sum_{i\in R^{\nu}} c_i R^{\nu}_i & \\
\text{s.t.} & ST_l - ST_j \geq \sum_{m\in M_j} b_{jlm}x_{jm} & (<j,l>\in E)\\
& ST_{n+1} \leq T & \\
& \sum_{j\in V(t)}\sum_{m\in M_j} r^{\rho}_{ijm}x_{jm} \leq R^{\rho}_i & (i\in R^{\rho}, t=0,\ldots,\overline{T}-1)\\
& \sum_{j\in V}\sum_{m\in M_j} r^{\nu}_{ijm}x_{jm} \leq R^{\nu}_i & (i\in R^{\nu})\\
& \sum_{m\in M_j} x_{jm} = 1 & (j\in V)\\
& x_{jm}\in\{0,1\} & (j\in V, m\in M_j)
\end{cases}
$$

The only difference between MRIP / max and MRLP / max is made by the objective function.

2.2 Basic Definitions and Theorems

In this section, we provide some basic definitions and results for digraphs which are used in Section 3 for the generation of cycle structures. For an introduction to the theory of graphs and digraphs we refer to Bondy and Murty (1976), Berge (1985), or Neumann and Morlock (1993).

We introduce the following notation. The symbols refer to digraph $\dot{G}=<V,E>$ or to network $\dot{G}=<V,E;c>$, repectively:

$\boldsymbol{A}$	adjacency matrix
C	set of cycle structures
$\dot{C}(i)$	cycle structure to which node $i \in V$ belongs; not defined, if $i \in V$ is not in the set of nodes of a cycle structure $\dot{C} \in C$
b_{ij}	weight of arc $<i,j>$
$\delta^-(i)$	outdegree of node $i \in V$
$\delta^+(i)$	indegree of node $i \in V$
E	set of arcs
$E(\dot{G})$	set of arcs of digraph or network $\dot{G}$, respectively
$\boldsymbol{I}$	identity matrix
$<i,j>$	arc from node $i \in V$ to node $j \in V$
$P(i)$	set of direct predecessors of node $i \in V$
R	set of sources
$\boldsymbol{R}$	reachability matrix
$R(i)$	set of nodes $j \in V$ which are reachable from node $i \in V$
$\bar{R}(i)$	set of nodes $j \in V$ from which node $i \in V$ can be reached
S	set of sinks
$S(i)$	set of direct successors of node $i \in V$
V	set of nodes
$V(\dot{G})$	set of nodes of digraph or network $\dot{G}$, respectively

We assume that digraph or network $\dot{G}$ is simple, that is, it contains no parallel arcs or directed loops.

Definition 1. Adjancency matrix $\boldsymbol{A}$ of a digraph

The adjacency matrix $\boldsymbol{A}$ of digraph $\dot{G}=<V,E>$ is defined to be the $|V|\times|V|$ matrix $\left(a_{ij}\right)_{i,j\in V}$ with $a_{ij}:=\begin{cases}1, \text{ if } <i,j>\in E\\ 0, \text{ otherwise}\end{cases}$.

Definition 2. Indegree and outdegree of node $i \in V$

The indegree $\delta^-(i)$ of node $i \in V$ is defined to be the number of (direct) predecessors of node i: $\delta^-(i):=|P(i)|$.
Analogously, the outdegree $\delta^+(i)$ of node $i \in V$ is defined to be the number of (direct) successors of node i: $\delta^+(i):=|S(i)|$.

Definition 3. Reachability

A node $j \in V$ is called reachable from node *i* if *j* = *i* or if there is a (directed) path W_{ij} with origin *i* and terminus *j*.

Definition 4. Reachability matrix of a digraph

The reachability matrix $\boldsymbol{R}$ of digraph $\dot{G} =< V,E >$ is defined to be the $n \times n$ matrix $\left(r_{ij}\right)_{i,j \in V}$ with $r_{ij} := \begin{cases} 1, \text{ if } j \text{ is reachable from } i \\ 0, \text{ otherwise} \end{cases}$.

Definition 5. Connectivity

Let $\dot{G} =< V,E >$ be a digraph with reachability matrix $\boldsymbol{R}$. Two nodes $i,j \in V$ are called connected in $\dot{G}$ if $i = j$ or if there is a sequence $(i_0, i_1, ..., i_k)$ of nodes $i_s \in V$ $(s = 0,...,k)$ with $i_0 = i,\ i_k = j,$ and $\prod_{s=1}^{k} \left[1 - (1 - r_{i_{s-1},i_s})(1 - r_{i_s,i_{s-1}})\right] = 1.$

Definition 6. Subdigraph and subdigraph induced by node set

A digraph $\dot{G}' =< V',E' >$ represents a subdigraph of digraph $\dot{G} =< V,E >$ if $V' \subseteq V, E' \subseteq E$ and $(< i,j > \in E' \Rightarrow i,j \in V')$.
A digraph $\dot{G}'' =< V'',E'' >$ represents the (unique) subdigraph of digraph $\dot{G} =< V,E >$ induced by node set V'' if $V'' \subseteq V$ and $(< i,j > \in E'' \Leftrightarrow i,j \in V'', < i,j > \in E)$.

Definition 7. Weak component

A weak component $\dot{G}' =< V',E' >$ of $\dot{G}$ is defined to be a maximal subdigraph of $\dot{G}$ (with respect to $|V'|$) induced by node set V' for which all nodes $i,j \in V'$ are connected. A digraph $\dot{G}$ which constitutes a weak component of itself is called weakly connected.

Remark 1.

If there is a subdigraph $\dot{G}' =< V',E' >$ of digraph $\dot{G} =< V,E >$ with $V' \subset V$, such that $\dot{G}'$ is a weak component of $\dot{G}$, then $\dot{G}$ is no weak component.

Definition 8. Network

An arc-weighted digraph $\dot{G} =< V,E;b >$ with $c: E \to \mathbb{R}$ is called network if the underlying digraph $\dot{G} =< V,E >$ is weakly connected.

Definion 9. Strong component

A strong component $\dot{G}' =< V',E' >$ of $\dot{G}$ is defined to be a maximal subdigraph of $\dot{G}$ (with respect to $|V'|$) for which all nodes $i,j \in V'$ are mutuallay reachable. A digraph $\dot{G}$ which constitutes a strong component of itself is called strongly connected.

Remark 2.

Obviously, any strong component is a weak component, too.

Definition 10. Cycle structure

A cycle structure $\dot{C}(i) =< V',E' >$ of $\dot{G}$ is a strong component of $\dot{G}$ with $|V'| \geq 2$.

Remark 3.

A cycle structure $\dot{C} = \dot{C}(i) =< V',E' >$ is the subdigraph of $\dot{G}$ induced by the node set $V' = \bar{R}(i) \cap R(i)$ with $\{i\} \subset V'$.

Definition 11. Contraction of a cycle structure.

The contraction of a cycle structure $\dot{C}$ to a contracted cycle structure c in a digraph $\dot{G} =< V,E >$ is an operation on $\dot{G}$ which derives a digraph $\dot{G}' =< V',E' >$ such that

$$(i) \quad V' := V \setminus V(\dot{C}) \cup \{c\}$$
$$E' := E \setminus \left\{ < i,j > \in E \middle| \{i,j\} \cap V(\dot{C}) \neq \emptyset \right\}$$
$$(ii) \quad \cup \left\{ < c,j > \middle| \exists < i,j > \in E : i \in V(\overset{r}{C}) \right\}$$
$$\cup \left\{ < i,c > \middle| \exists < i,j > \in E : j \in V(\overset{r}{C}) \right\}$$

Definition 12. Expansion of a contracted cycle structure

Let c be a contracted cycle structure $\dot{C}$ in a digraph $\dot{G}=<V,E>$ and $\dot{G}'=<V',E'>$ a digraph with subdigraph $\dot{G}$. By the expansion of c with respect to $\dot{G}'$ we mean an operation on $\dot{G}$ which derives a digraph $\dot{G}''=<V'',E''>$ such that

(*i*) $V'':=V\cup(V(\dot{C})\cap V')\setminus\{c\}$

(*ii*) $E'':=\left\{<i,j>\in E'\middle|\{i,j\}\subseteq V''\right\}\setminus\left\{<i,j>\in E\middle|c\in\{i,j\}\right\}$.

Definition 13. Redundant arc

An arc $<i,j>\in E$ is said to be redundant in $\dot{G}=<V,E>$ if there is a (directed) path W_{ij} in $\dot{G}$ which contains more than one arc.

Remark 4.

Obviously, it holds that: $<i,j>$ redundant $\Leftrightarrow r'_{ij}=1$ for $\dot{G}'=<V,E\setminus\{<i,j>\}>$ with reachability matrix $\boldsymbol{R}'$.

Definition 14. Redundancy-generating arc

An arc $<i,j>\in E$ is said to be redundancy-generating in $\dot{G}=<V,E>$ if $<i,j>$ is redundant in $\dot{G}$ or if there is an arc $<k,l>\in E$ and a (directed) path W_{kl} in $\dot{G}$ such that $<i,j>$ belongs to W_{kl}.

Remark 5 (cf. Kolisch et al. 1995).

$<i,j>$ is redundancy-generating in $\dot{G}\Leftrightarrow$ one of the following two cases is true

(*i*) $j\in R(i)\setminus S(i)$

(*ii*) $\exists l\in R(j): P(l)\cap\bar{R}(i)\neq\emptyset$

Theorem 1.

Let $\dot{G}=<V,E>$ be an acyclic digraph with reachability matrix $\boldsymbol{R}$ and $<i,j>\notin E$, and let

$$(2.5)\quad \rho_{ij}:=\delta\left(\sum_{k\in\bar{R}(i)}\sum_{l\in R(j)}a_{kl}\right)\quad (i,j\in V)\ \text{with}\ \delta(x):=\begin{cases}1, & \text{if } x>0\\ 0, & \text{otherwise}\end{cases}.$$

Then, the following equivalences hold:

(*i*) $j \in R(i) \setminus S(i) \Leftrightarrow r_{ij} = 1$

(*ii*) $(\exists l \in R(j): P(l) \cap \bar{R}(i) \neq \emptyset) \Leftrightarrow \rho_{ij} = 1$

Proof. Obvious.

Corollary 1.

Let $\dot{G} =< V,E >$ be an acyclic digraph with $< i,j > \notin E$. The insertion of arc $< i,j >$ generates redundancy (that is, $< i,j >$ is redundancy-generating in $\dot{G}' =< V, E \cup \{< i,j >\} >$) exactly if $r_{ij} + \rho_{ij} > 0$.

Theorem 2.

The maximum number m_{red}^{max} of arcs $< i,j >$ which can be inserted in an acyclic digraph $\dot{G} =< V,E >$ without redundant arcs, such that $< i,j >$ is redundant in digraph $\dot{G}' =< V, E \cup \{< i,j >\} >$ and $\dot{G}'$ is acyclic is

(2.6) $$m_{red}^{max} = \sum_{i,j \in V} r_{ij} - |V| - |E|.$$

Proof.

With Remark 4 and

(*i*) $r_{ii} = 1 \; \forall i \in V$ and

(*ii*) $< i,j > \in E \Rightarrow r_{ij} = 1$

we obtain

(2.7) $$m_{red}^{max} = \sum_{i \in V} \sum_{\substack{j \in V \setminus \{i\} \\ < i,j > \notin E}} r_{ij} = \sum_{i,j \in V} r_{ij} - \sum_{i \in V} r_{ii} - \sum_{< i,j > \in E} r_{ij} = \sum_{i,j \in V} r_{ij} - |V| - |E|.$$ ❑

Theorem 3.

Let $\dot{G} =< V,E >$ be a digraph with reachability matrix $\boldsymbol{R}$. Let $\boldsymbol{R}^2 := \left(r_{ij}^{(2)}\right)_{i,j \in V}$ be the squared reachability matrix. If $r_{ij} = 1$, the cardinality of node set V' of the cycle structure $\dot{C}(i) = \dot{C}(j)$ which results from the insertion of arc $< j,i >$ in $\dot{G}$ is

(2.8) $|V'| = \left|V(\dot{C}(i))\right| = \left|V(\dot{C}(j))\right| = r_{ij}^{(2)}$.

Proof.

Due to $r_{ij} = 1$, the insertion of arc $< j,i >$ generates the cycle structure $\dot{C}(i) = \dot{C}(j)$ with node set $R(i) \cap \bar{R}(i) = R(j) \cap \bar{R}(j)$.

(*i*) $r_{ij}^{(2)} \leq |V'|$:

$$r_{ij}^{(2)} = \sum_{k \in V} r_{ik} r_{kj} = \left|\left\{l \in V \middle| l \in R(i) \cap \bar{R}(j)\right\}\right|.$$

After the insertion of $< j,i >$ we obtain: $k \in R(j) \cap \bar{R}(i)\ \forall k \in \left\{l \in V \middle| l \in R(i) \cap \bar{R}(j)\right\}$.

$\Rightarrow k \in R(i) \cap \bar{R}(i) = R(j) \cap \bar{R}(j) = V(\dot{C}(i))\ \forall k \in \left\{l \in V \middle| l \in R(i) \cap \bar{R}(j)\right\}$

$\Rightarrow \left\{l \in V \middle| l \in R(i) \cap \bar{R}(j)\right\} \subseteq V(\dot{C}(i))$

$\Rightarrow r_{ij}^{(2)} \leq \left|V(\dot{C}(i))\right| = |V'|$

(*ii*) $r_{ij}^{(2)} \geq |V'|$

After the insertion of $< j,i >$, let $k \in V$ be a node of the cycle structure $\dot{C}(i)$. Then, *k* is a node on a path W_{ij} from node *i* to node *j*.

$\Rightarrow \forall k \in V(\dot{C}(i))\text{: } r_{ik} r_{kj} = 1$

$\Rightarrow \sum_{k \in V(\dot{C}(i))} r_{ik} r_{kj} = \left|V(\dot{C}(i))\right|$

$\Rightarrow r_{ij}^{(2)} \geq \left|V(\dot{C}(i))\right| = |V'|$

(*i*), (*ii*) $\Rightarrow r_{ij}^{(2)} = |V'|$ ❑

Corollary 2.

Let $\dot{G} =< V,E >$ be a digraph with squared reachability matrix $\boldsymbol{R}^2$ and set of cycle structures C. Then,

(2.9) $\left|V(\dot{C}(i))\right| = r_{ii}^{(2)}\ \forall i \in \bigcup_{\dot{C} \in C} V(\dot{C})$.

Corollary 3.

Let $\dot{G}=<V,E>$ be a digraph with squared reachability matrix $\boldsymbol{R}^2$ and set of cycle structures C. The number $\Gamma:=|C|$ of cycle structures in $\dot{G}$ is

(2.10) $$\Gamma = \sum_{\substack{i\in V \\ r_{ii}^{(2)}>1}} \frac{1}{r_{ii}^{(2)}}.$$

Remark 6.

For $i \neq j$ $r_{ij}^{(2)} \leq 1$ implies $r_{ij} = 0$, since
$r_{ij}^{(2)} = \sum_{k\in V} r_{ik}r_{kj} = \sum_{k\in V\setminus\{i,j\}} r_{ik}r_{kj} + r_{ii}r_{ij} + r_{ij}r_{jj} \geq r_{ii}r_{ij} + r_{ij}r_{jj} = 2r_{ij} > 1$ if $r_{ij} = 1$.

Remark 7.

Let $\dot{G}=<V,E>$ be a digraph with adjacency matrix $\boldsymbol{A}$ and the squared reachability matrix $\boldsymbol{R}^2$. Arc $<i,j>\in E$ is redundant in $\dot{G}$ exactly if $a_{ij}=1$ and $r_{ij}^{(2)} > 2$.

Definition 15. Creation of a cycle structure within a digraph

Let $\dot{G}=<V,E>$ be a digraph with set of cycle structures C. By a creation of a cycle structure $\dot{C}(i)$ within $\dot{G}$ we mean an operation on $\dot{G}$ which derives a digraph $\dot{G}'=<V,E'>$ with set of cycle structures C' such that $C'\supset C$, $E'\supset E$, and $|E'|=|E|+1$.

Definition 16. Extension of a cycle structure within a digraph

Let $\dot{G}=<V,E>$ be a cyclic digraph with set of cycle structures $C\neq\emptyset$. By an extension of a cycle structure $\dot{C}(i)\in C$ within $\dot{G}$ we mean an operation on $\dot{G}$ which derives a digraph $\dot{G}'=<V,E'>$ with set of cycle structures C' such that $|C|=|C'|$, $E'\supset E$, $|E'|=|E|+1$, and $\exists\dot{C}'(i)\in C'$ with $V(\dot{C}'(i))\supset V(\dot{C}(i))$.

Definition 17. Densification of a cycle structure within a digraph $\dot{G}$

Let $\dot{G}=<V,E>$ be a cyclic digraph with set of cycle structures $C\neq\emptyset$. By a densification of a cycle structure $\dot{C}\in C$ within $\dot{G}$ we mean an operation on $\dot{G}$ which derives a digraph

$\dot{G}'=<V,E'>$ with set of cycle structures C' such that $|C|=|C'|$, $E' \supset E$, $|E'|=|E|+1$, and $V(\dot{C}'(i)) = V(\dot{C}(i)) \; \forall \dot{C}'(i) \in C'$.

Definition 18. Acyclic skeleton of a digraph

Let $\dot{G}'=<V,E'>$ be a digraph with reachability matrix $\boldsymbol{R}'$. We call digraph $\dot{G}=<V,E>$ with reachability matrix $\boldsymbol{R}$ an acyclic skeleton of $\dot{G}'$ if

(*i*) $E \subseteq E'$
(*ii*) $r_{ij}' r_{ji}' = 0 \Rightarrow r_{ij} = r_{ij}'$
(*iii*) $r_{ij}' r_{ji}' = 1 \Rightarrow r_{ij} + r_{ji} = 1$
(*iv*) $r_{ij} = 1, <i,j> \in E' \Rightarrow <i,j> \in E$

Theorem 4.

Let $\dot{G}=<V,E>$ be an acyclic digraph. Then, any cyclic digraph $\dot{G}'=<V,E'>$ for which $\dot{G}$ is an acyclic skeleton can be obtained by first performing creations of cycle structures within $\dot{G}$, then performing extensions of cycle structures within $\dot{G}$, and finally performing densifications of cycle structures within $\dot{G}$.

Proof.

Let $\dot{G}=<V,E>$ with reachability matrix $\boldsymbol{R}$ be an acyclic skeleton of digraph $\dot{G}'=<V,E'>$ with reachability matrix $\boldsymbol{R}'$ and set of cycle structures C'. The following algorithm transforms $\dot{G}$ to a digraph $\dot{G}''=<V,E''>$: We will show that Step 1 corresponds to the creation of cycle structures, Step 2 extends cycle structures, Step 3 densifies cycle structures and that digraph $\dot{G}''$ which is obtained at the end of Step 3 equals $\dot{G}'$.

(0) Initialization
$E'':=E, \boldsymbol{R}'':=\boldsymbol{R}, C'':=\varnothing$.

(1) Creation of cycle structures
WITH $\dot{C}' \in C'$ DO
Determine an arc $<i,j> \in (E' \setminus E'') \cap E(\dot{C}')$.
$E'':=E'' \cup \{<i,j>\}$. Update matrix $\boldsymbol{R}''$ and set C''.
END (* WITH *).

(2) Extension of cycle structures

WHILE $\exists < i,j > \in E' \setminus E'' : (r_{ij}'' = 0 \wedge \{i,j\} \cap \underset{C'' \in C''}{\mathrm{r}\bigcup} V(\dot{C}'') \neq \emptyset)$ DO

Select $< i,j > \in E' \setminus E''$ with $r_{ij}'' = 0 \wedge \{i,j\} \cap \underset{C'' \in C''}{\mathrm{r}\bigcup} V(\dot{C}'') \neq \emptyset$.

$E'' := E'' \cup \{< i,j >\}$. Update matrix $\boldsymbol{R}''$ and set C''.

END (* WHILE *).

(3) Densification of cycle structures

WITH $< i,j > \in E' \setminus E''$ DO

$E'' := E'' \cup \{< i,j >\}$.

END (* WITH *).

For $< i,j > \in E' \setminus E$ it follows from (*iv*) that $r_{ij} = 0$. With $r_{ij}' = 1$ and (*ii*) we obtain $r_{ij}' r_{ji}' = 1$, that is, $< i,j > \in E' \setminus E \Rightarrow r_{ij}' r_{ji}' = 1$. With (*iii*) we obtain $< i,j > \in E' \setminus E \Rightarrow r_{ji} = 1$.

That is the reason why the arcs $< i,j > \in E' \setminus E$ which are added to E'' in Steps 1, 2, and 3 belong to one of the cycle structure $\dot{C}'$ of $\dot{G}'$ ($< i,j > \in E(\dot{C}')$). From $r_{ij} = 0$, $r_{ji} = 1$, and (*i*) it can be concluded that any arc added to E'' in Step 1 generates an *additional* cycle structure in $\dot{G}''$. Obviously, after Step 1 we have $|C''| = |C'|$. Due to (*i*) and $|C''| = |C'|$, no cycle structure will be generated in Step 2 or Step 3.

Since in Step 1 no arc has been removed from E'', we have $r_{ij} = 0$ for any arc $< i,j > \in E' \setminus E''$ with $r_{ij}'' = 0$ after Step 1. Hence, $r_{ij}'' = 0 \Rightarrow r_{ij} = 0 \Rightarrow r_{ji}' = 1$ for $< i,j > \in E' \setminus E''$ and the addition of arcs $< i,j >$ in Step 2 extends a cycle structure $\dot{C}'' \in C''$.

After Step 2, it holds that for any cycle structure $\dot{C}' \in C'$ there is a cycle structure $\dot{C}'' \in C''$ with $V(\dot{C}'') = V(\dot{C}')$ since, otherwise, there would be a cycle structure $\dot{C}' \in C'$ and nodes $j,l \in V(\dot{C}')$ with $r_{jl} = 0$ which obviously contradicts the strong connectivity of $\dot{C}'$. Hence, $r_{ij}' r_{ji}' = 1 \Leftrightarrow r_{ij}'' r_{ji}'' = 1$. With $< i,j > \in E' \setminus E'' \Rightarrow < i,j > \in E' \setminus E \Rightarrow r_{ij}' r_{ji}' = 1 \Rightarrow r_{ij}' r_{ji}' = 1$ we have shown that Step 3 corresponds to the densification of cycle structures. Due to (*i*), $E'' = E'$ after Step 3. ❑

Lemma 1.

Let $\dot{G}$ be an acyclic weak component with (at least) two sinks (that is, $|S| \geq 2$). Then, for any sink $s \in S$, there are a further sink $s' \in S, s' \neq s$ and a source $r \in R$, such that $r \in \bar{R}(s) \cap \bar{R}(s')$.

Proof.

Since $\dot{G}$ is acyclic, for each sink $s \in S$ there is a source $r \in R$, such that $r \in \bar{R}(s)$.

Let us assume that for a given sink s there are no sink $s' \in S, s' \neq s$ and source $r \in R$, such that $r \in \bar{R}(s) \cap \bar{R}(s')$.

$\Rightarrow \bar{R}(s) \cap \bar{R}(s') = \emptyset \; \forall s' \in S: s' \neq s$
$\Rightarrow$ the subdigraph $\dot{G}'$ of $\dot{G}$ induced by $\bar{R}(s)$ is a weak component of $\dot{G}$, and $\dot{G}' \neq \dot{G}$ since $s' \notin \bar{R}(s)$.
$\Rightarrow \dot{G}$ is not weakly connected, which contradicts the assumptions. ❑

Lemma 2.

Let $\dot{G}$ be an acyclic weak component with (at least) two sources (that is, $|R| \geq 2$). Then, for any source $r \in R$, there are a further source $r' \in R, r' \neq r$ and a sink $s \in S$, such that $s \in R(r) \cap R(r')$.

Proof. The proof can be led analogously to Lemma 1.

Theorem 5.

Let $\dot{G}$ be an acyclic weak component with two sources and two sinks (that is, $|R| \geq 2$, $|S| \geq 2$). Then, there is always a sink $s \in S$ with $|\bar{R}(s) \cap R| \geq 2$, and for each sink $s \in S$ with $|\bar{R}(s) \cap R| \geq 2$ there are two corresponding sources $r, r' \in R, r \neq r'$ with $s \in R(r) \cap R(r')$, where r can be chosen such that there is a sink $s' \in S, s' \neq s$ with $r \in \bar{R}(s) \cap \bar{R}(s')$.

Proof.

Since $\dot{G}$ is acyclic and weakly connected with $|R| \geq 2$ and $|S| \geq 2$, the assumptions of Lemmata 1 and 2 are met.
From Lemma 2 it follows that $\exists s \in S: (\exists r_1, r_2 \in R, r_1 \neq r_2: s \in \bar{R}(r_1) \cap \bar{R}(r_2))$.
With Lemma 1 we obtain that there is a further sink $s' \in S, s' \neq s$ such that there is a source $\hat{r} \in R$ with $\hat{r} \in \bar{R}(s) \cap \bar{R}(s')$.

(*i*) $\hat{r} \neq r_1, \hat{r} \neq r_2$

$\Rightarrow r_1, r_2, \hat{r} \in \bar{R}(s)$ and $\hat{r} \in \bar{R}(s')$.
By setting $r := \hat{r}, r' := r_1$ or $r' := r_2$, we obtain $s \in R(r) \cap R(r')$ and $r \in \bar{R}(s) \cap \bar{R}(s')$.

(*ii*) $\hat{r} = r_1$

$\Rightarrow r_2, \hat{r} \in \bar{R}(s)$ and $\hat{r} \in \bar{R}(s')$.
By setting $r := \hat{r}, r' := r_2$, we obtain $s \in R(r) \cap R(r')$ and $r \in \bar{R}(s) \cap \bar{R}(s')$.

(*iii*) $\hat{r} = r_2$

$\Rightarrow r_1, \hat{r} \in \bar{R}(s)$ and $\hat{r} \in \bar{R}(s')$.
By setting $r := \hat{r}, r' := r_1$, we obtain $s \in R(r) \cap R(r')$ and $r \in \bar{R}(s) \cap \bar{R}(s')$. ❑

Corollary 4.

Let $\dot{G}$ be an acyclic weak component with two sources and two sinks. Then, there are sources $r,r' \in R, r \neq r'$ and sinks $s,s' \in S, s \neq s'$ such that $r \in \bar{R}(s) \cap \bar{R}(s')$ and $s \in R(r')$.

Theorem 6.

Let $\dot{G}$ be an acyclic weak component with two sources and two sinks. Let $r,r' \in R, r \neq r'$ and $s,s' \in S, s \neq s'$ with $r \in \bar{R}(s) \cap \bar{R}(s')$ and $s \in R(r')$.
Then, the insertion of arc $< s,r >$ generates a cycle structure $\dot{C}(r) = \dot{C}(s)$ such that the contraction of $\dot{C}(r) = \dot{C}(s)$ to the contracted cycle structure c in $\dot{G}$ derives a digraph $\dot{G}'$ in which c neither constitutes a source nor a sink.

Proof.

From Corollary 3 it follows that there are sources $r,r' \in R, r \neq r'$ and sinks $s,s' \in S, s \neq s'$ with $r \in \bar{R}(s) \cap \bar{R}(s')$ and $s \in R(r')$. We obtain:

(i) $\quad r \in \bar{R}(s), s' \notin V(\dot{C}(r)) \Rightarrow c \in \bar{R}(s')$

(ii) $\quad s \in R(r'), r' \notin V(\dot{C}(r)) \Rightarrow c \in R(r')$

(i), (ii) $\Rightarrow s' \in R(c)$ and $r' \in \bar{R}(c)$ which implies $R(c) \setminus \{c\} \neq \emptyset$ and $\bar{R}(c) \setminus \{c\} \neq \emptyset$. Hence, c neither constitutes a sink nor a source. ❑

2.3 Network Measures

The structure of the underlying network generally has a strong impact on the time which an exact algorithm requires for solving a sequencing problem as well as on the gap between the objective function values of solutions which have been obtained by heuristics and the objective function value of an optimum.
In literature, a large number of network measures can be found which describe the size, the logic, and the shape of networks (cf. Thesen 1977, Elmaghraby and Herroelen 1980, Davis 1975, Kaiman 1974, Kurtulus and Davis 1982, Patterson 1976).
Table 1 summarizes the control parameters used by ProGen/max for the generation of the structure $\dot{G} =< V,E >$ of project network $\dot{N} =< V,E;b >$:

Measure	Definition
Number of nodes	$\|V\|$
Thesen's estimator for the restrictiveness (see below)	$RT = \frac{2\sum_{i,j\in V} r_{ij} - 6(\|V\|-1)}{(\|V\|-2)(\|V\|-3)}$
Degree of redundancy	$\rho = \frac{\left\|\left\{< i,j >\in E \middle\| r_{ij}^{(2)} > 2\right\}\right\|}{m_{red}^{max}}$
Number of predecessors and number of successors of a node	$\|P(i)\|, \|S(i)\| \; (i \in V)$
Number of cycle structures	$\Gamma := \|C\| = \sum_{\substack{i\in V \\ r_{ii}^{(2)}>1}} \frac{1}{r_{ii}^{(2)}}$
Number of backward arcs	$\left\|\left\{< j,i >\in E \middle\| r_{ij} = 1\right\}\right\|$
Number of nodes in a cycle structure	$r_{ii}^{(2)} \; (i \in \bigcup_{C\in C} V(\dot{C}))$

Table 1. Network measures

Instead of the most commonly used *CNC* coefficient of network complexity (that is, the ratio of the number of arcs to the number of nodes), we employ one of the approximations for the restrictiveness devised by Thesen (1977).

Definition 19: Restrictiveness of a digraph

Let $\dot{G} =< V,E >$ be a weakly connected digraph with exactly one source 0 and exactly one sink $n+1$ and node set $V = \{0,1,...,n,n+1\}$. Let Π denote the number of permutations $(i_1, i_2, ..., i_n)$ of $V' = \{1,...,n\} \subseteq V$ such that $k < l \Rightarrow i_k \notin R(i_l)$. The restrictiveness is defined as $\mathrm{P} := 1 - \frac{\log \Pi}{\log n!}$.

Remark 8.

$\mathrm{P} \in [0,1]$, $\mathrm{P} = 0$ for parallel digraphs, and $\mathrm{P} = 1$ for series digraphs (cf. Thesen 1977).

P represents an exact measure of the degree to which precedence constraints restrict the number of feasible node sequences. Thus, P is an appropriate index of network complexity. The determination of Π, however, constitutes a hard combinatorial problem. That is why Thesen has tested a set of over 40 different estimators for P. With *RT* we denote that estimator which yields the lowest mean relative error (with respect to P) in the empirical analysis of Thesen:

Definition 20. Estimator *RT* for the restrictiveness

Let $\dot{G} =< V, E >$ be a weakly connected acyclic digraph with exactly one source 0 and exactly one sink $n+1$, node set $V = \{0,1,...,n,n+1\}$ and reachability matrix $\boldsymbol{R}$. Fictitious (undirected) edges with incident nodes $i, j \in V$ between which no precedence relation has been established (that is, $i \notin \bar{R}(j) \cup R(j)$) are called disjunctive arcs. Let n_d be the number of disjunctive arcs in digraph $\dot{G}$ and let n_d^{max} be the maximum number of possible disjunctive arcs in a weakly connected digraph with node set *V*, exactly one source, and exactly one sink. Then, the restrictiveness estimator *RT* is defined to be

$$RT := 1 - \frac{n_d}{n_d^{max}}. \tag{2.11}$$

Theorem 7.

Let $\dot{G} =< V, E >$ be a weakly connected acyclic digraph with exactly one source 0 and exactly one sink $n+1$, node set $V = \{0,1,...,n,n+1\}$, reachability matrix $\boldsymbol{R}$, and restrictiveness estimator *RT*. Then,

$$RT = 1 - \frac{(n+2)(n+3) - 2\sum_{i,j \in V} r_{ij}}{n(n-1)}. \tag{2.12}$$

Proof.

There is a disjunctive arc between nodes $i, j \in V$ exactly if $r_{ij} + r_{ji} = 0$. Moreover, $r_{ij} + r_{ji} = 0 \Leftrightarrow r_{ij} + r_{ji} - r_{ij} r_{ji} \neq 1$. Since $\dot{G}$ is acyclic by assumption, we have $r_{ij} r_{ji} = 0$ for $i \neq j$. For the number of disjunctive arcs n_d we obtain

$$\begin{aligned} n_d &= \frac{1}{2}\sum_{i \in V}\left[(n+2) - \sum_{j \in V}(r_{ij} + r_{ji} - r_{ij} r_{ji})\right] \\ &= \frac{1}{2}\sum_{i \in V}\left[(n+2) - \sum_{j \in V}(r_{ij} + r_{ji}) + 1\right] \\ &= \frac{1}{2}\left[(n+2)^2 + n + 2 - \sum_{i,j \in V}(r_{ij} + r_{ji})\right] \\ &= \frac{1}{2}\left[(n+2)(n+3) - 2\sum_{i,j \in V} r_{ij}\right]. \end{aligned} \tag{2.13}$$

Note that $r_{ii} = 1 \; \forall i \in V, r_{0j} = 1 \; \forall j \in V, r_{i,n+1} = 1 \; \forall i \in V$. From (2.13) it follows that the number of disjunctive arcs is maximal if $r_{ij} = 0 \; \forall i, j \in V \setminus \{0, n+1\}, i \neq j$. Hence, we obtain for the maximum number of disjunctive arcs $n_d^{max} = \frac{1}{2}\left[(n+2)(n+3) - 2\left[3(n+2)-3\right]\right] = \frac{n(n-1)}{2}$.

Then, $RT = 1 - \frac{n_d}{n_d^{max}} = 1 - \frac{\frac{1}{2}\left[(n+2)(n+3) - \sum_{i,j \in V} r_{ij}\right]}{n(n-1)/2} = 1 - \frac{(n+2)(n+3) - 2\sum_{i,j \in V} r_{ij}}{n(n-1)}$.

The computational results obtained by Thesen for the accuracy of RT w.r.t. the restrictiveness P of acyclic digraphs will be confirmed in Section 5.1. The properties of RT stated in the following theorem could explain the very good performance of RT in predicting the restrictiveness of networks.

Theorem 8.

Let $\dot{G} =< V, E >$ be a weakly connected acyclic digraph with exactly one source 0 and exactly one sink $n+1$, node set $V = \{0, 1, \ldots, n, n+1\}$, reachability matrix $\boldsymbol{R}$, and restrictiveness estimator RT. For RT the following properties apply:

(*i*) $RT \in [0,1]$.

(*ii*) RT=0 exactly if $\dot{G}$ is parallel.

(*iii*) RT=1 exactly if $\dot{G}$ is serial.

(*iv*) The insertion of a non-redundant arc in $\dot{G}$ increases RT.

(*v*) The insertion of a redundant arc in $\dot{G}$ does not affect RT.

Proof.

From the weak connectivity of $\dot{G}$ it follows that $r_{0j} = 1 \; \forall j \in V$, $r_{i,n+1} = 1 \; \forall i \in V$, $r_{i0} = 0 \; \forall i \in V$, and $r_{n+1,j} = 0 \; \forall j \in V$.

(*i*)

Since $\dot{G}$ is acyclic, it holds that $3n+3 \leq \sum_{i,j \in V} r_{ij} \leq \frac{(n+2)(n+3)}{2}$. For RT we obtain

$$\frac{(n+2)(n+3) - (n+2)(n+3)}{n(n-1)} = 0 \leq 1 - RT \leq 1 = \frac{(n+2)(n+3) - 6n - 6}{n(n-1)}.$$

(*ii*) parallel digraphs:

$\dot{G}$ is a parallel digraph $\Leftrightarrow r_{ij} = 0 \; \forall i, j \in V \setminus \{0, n+1\}, i \neq j$. We obtain:

$$RT = 1 - \frac{(n+2)(n+3) - 2\sum_{i,j \in V} r_{ij}}{n(n-1)} = 1 - \frac{(n+2)(n+3) - 2\left(\sum_{i,j \in V \setminus R \setminus S} r_{ij} + \sum_{i \in R, j \in V} r_{ij} + \sum_{i \in V \setminus R, j \in S} r_{ij}\right)}{n(n-1)}$$

$$= 1 - \frac{(n+2)(n+3) - 2(n+n+2+n+1)}{n(n-1)}$$

$$= 1 - \frac{(n+2)(n+3) - 6n - 6}{n(n-1)} = 1 - \frac{n^2 - n}{n(n-1)} = 1 - 1 = 0$$

(*iii*) series digraphs:

$\dot{G}$ is a series digraph $\Leftrightarrow$ there is a permutation $(0, j_1, \ldots, j_n, n+1)$ of the nodes $j_\nu \in \{1, \ldots, n\}$ such that $r_{i_\mu j_\nu} = 1 \Leftrightarrow \mu \leq \nu$. It can easily be shown, that in that case $\sum_{i,j \in V} r_{ij} = \frac{|V|(|V|+1)}{2} = \frac{(n+2)(n+3)}{2}$. For RT we obtain:

$$RT = 1 - \frac{(n+2)(n+3) - 2\sum_{i,j \in V} r_{ij}}{n(n-1)} = 1 - \frac{(n+2)(n+3) - 2\frac{1}{2}(n+2)(n+3)}{n(n-1)} = 1$$

(*iv*)

Obviously, the insertion of any arc $< i, j > (i, j \in V)$ in $\dot{G}$ cannot decrease $\sum_{i,j \in V} r_{ij}$.

If $< i, j >$ is not redundant, at least r_{ij} will be set from 0 to 1, which increases $\sum_{i,j \in V} r_{ij}$.

(*v*)

If $< i, j >$ is redundant, there is a (directed) path W_{ij} from i to j which includes more than one arc. In that case, there will be no $k, l \in V$ such that r_{kl} is set from 0 to 1. ❑

Remark 9.

Let $ST := (n!)^{1-RT}$ be the estimator of Π based on RT. For cases (*ii*) and (*iii*) of Theorem 8, ST represents the exact number of feasible permutations Π. Properties (*iv*) and (*v*) give a (partial) explanation for the good behaviour of estimator ST even for digraphs which are not parallel or series, since the insertion of non-redundant arcs always decreases the num-

ber of feasible permutations, whereas the introduction of additional redundant arcs does not influence the precedence constraints.

Because of the good approximation of the number of feasible activity sequences, ST probably has a strong impact on the hardness of instances of project scheduling problems. Recently, de Reyck and Herroelen (1994) investigated the relationship between the hardness of problem instances and the reduction complexity index CI of the underlying project network for RCPSP and the time/cost tradeoff problem. In de Reyck (1995) it is shown that the more intuitive measure RT plays an even more important role for the computational effort required to solve instances of RCPSP. The empirical analysis presented in Section 5.2 will show the strong impact of RT on the hardness of RCPSP/max instances.

3. Generation of the Basic Data and the Network

3.1 Generation of the Basic Data

The user of ProGen/max has to enter values for the following basic data which will be used for the construction of the project network and the generation of the resource data:

n^{min}, n^{max}	minimal and maximal number of activities
M^{min}, M^{max}	minimal and maximal number of modes per activity
R^{min}, R^{max}	minimal and maximal number of resources
$p_\rho \in [0,1],\ p_\nu \in [0,1]$	percentage of renewable and nonrenewable resources, respectively (the percentage of doubly-constrained resources is $p_\delta := 1 - p_\rho - p_\nu$)
$c^{\rho,min}, c^{\rho,max}$	minimal and maximal costs for the period availability of one unit of a renewable or doubly-constrained resources
$c^{\nu,min}, c^{\nu,max}$	minimal and maximal costs for the total availability of one unit of a nonrenewable or doubly-constrained resources

Let $rand\{a,\ldots,b\}\ (a,b \in \mathrm{IN}_0)$ be an integer pseudo random number out of the set $\{a,\ldots,b\}$ and let $rand[a,b]\ (a,b \in \mathrm{IR})$ be a real pseudo random number out of the interval [*a*,*b*], both based on a uniformly distributed pseudo random number generated with the congruence-generator of Schrage (cf. Schrage 1979). With $\mathrm{int}(x)\ (x \geq 0)$ we denote the rounded value of x: $\mathrm{int}(x) := \lfloor x + 0.5 \rfloor$.

The basic data is calculated as follows:

- number of activities: $n := rand\left\{n^{min},\ldots,n^{max}\right\}$
- number of modes of activity j: $\left|M_j\right| := rand\left\{M^{min},\ldots,M^{max}\right\}\ (j \in V \setminus \{0, n+1\})$
- number of resources: $\left|R^\rho \cup R^\nu\right| := rand\left\{R^{min},\ldots,R^{max}\right\}$
- number of renewable resources: $\left|R^\rho \setminus R^\nu\right| := \mathrm{int}(p_\rho \left|R^\rho \cup R^\nu\right|)$

- number of nonrenewable resources: $\left|R^{\nu} \setminus R^{\rho}\right| := \mathrm{int}(p_{\nu}\left|R^{\rho} \cup R^{\nu}\right|)$
- number of doubly-constrained resources: $\left|R^{\nu} \cap R^{\rho}\right| := \mathrm{int}(p_{\delta}\left|R^{\rho} \cup R^{\nu}\right|)$
- cost coefficients: $c_i := rand\left\{c^{\rho,min},...,c^{\rho,max}\right\} (i \in R^{\rho})$ and $c_i := rand\left\{c^{\nu,min},...,c^{\nu,max}\right\} (i \in R^{\nu})$

3.2 Network Structure

Let $\dot{G} =< V,E >$ be the weakly connected digraph which represents the structure of the project network $\dot{N} =< V,E;b >$ under consideration. Precedence and time constraints of corresponding instances of problems MRCPSP/max, MRLP/max, and MRIP/max (cf. (2.1), (2.3), and (2.4)) are given by the digraph $\dot{G}$ and the corresponding minimal and maximal time lags.

Obviously, several minimal time lags ${}^{1}T_{ij}^{min},...,{}^{n_{ij}}T_{ij}^{min}$ between two activities *i* and *j* can be replaced by $T_{ij}^{min} := \max\limits_{\nu=1,...,n_{ij}} {}^{\nu}T_{ij}^{min}$, whereas several maximal time lags ${}^{1}T_{ij}^{max},...,{}^{n_{ij}}T_{ij}^{max}$ between two activities *i* and *j* can be replaced by $T_{ij}^{max} := \min\limits_{\nu=1,...,n_{ij}} {}^{\nu}T_{ij}^{max}$. A minimal time lag T_{ij}^{min} between the start of activity *i* and the start of activity *j* can be represented by an arc $< i,j >$ weighted by $b_{ij} := T_{ij}^{min}$. A maximal time lag T_{ij}^{max} between the start of activity *i* and the start of activity *j* can be represented by a (backward) arc $< j,i >$ weighted by $b_{ji} := -T_{ij}^{max}$. Negative minimal time lags can be considered as positive maximal time lags and vice versa (cf. Neumann and Schwindt 1995).

If there is a minimal time lag $T_{ij}^{min} > 0$ and a maximal time lag $T_{ji}^{max} > 0$ which both would be represented by an arc $< i,j >$, the maximal time lag T_{ji}^{max} can be neglected since it will always be met if we observe the minimal time lag T_{ij}^{min}. That is why there will be no parallel arcs in digraph $\dot{G}$ and the corresponding project network $\dot{N}$. Since a precedence or a time constraint concerning a single activity does not make sense in project scheduling, we can rule out the case of loops and the generation of the network structure can be limited to the case of simple digraphs.

We consider two methods for the generation of cyclic network structures.

The first algorithm, called *direct method* in the following, starts with the generation of an acyclic, generally not weakly connected digraph. Then, backward arcs are added to generate cycle structures. Finally, a supersource and a supersink are added to obtain a weakly connected digraph.

The second algorithm, called *contraction method* in the following, first creates cycle structures which are then contracted. With the contracted cycle structures and the nodes not employed during the first step we generate an acyclic, generally not weakly connected digraph, similarly to the direct method. Subsequently, the contracted cycle structures are expanded and integrated into the digraph. Finally, we add a supersource and a supersink to obtain a weakly connected digraph.

Algorithm A2. *Direct method*

(1) Generation of an acyclic digraph without redundancy

  (1.1) Selection of sources and sinks (nodes which will correspond to initial and terminal activities)
  (1.2) Generation of direct predecessors
  (1.3) Generation of direct successors
  (1.4) Insertion of additional arcs such that the resulting digraph is still without redundancy

(2) Insertion of redundant arcs

(3) Generation of cycle structures

  (3.1) Creation of cycle structures
  (3.2) Extension of cycle structures
  (3.3) Densification of cycle structures

(4) Addition of a supersource and a supersink

Algorithm A3. *Contraction method*

(1) Generation of cycle structures

  (1.1) Generation of several weak components
  (1.2) Transformation of the weak components to cycle structures

(2) Contraction of the cycle structures to contracted cycle structures

(3) Generation of an acyclic digraph based on the contracted cycle structures and additional nodes

(4) Expansion of the contracted cycle structures and insertion in the digraph

(5) Addition of a supersource and a supersink

Subsection 3.2.1 deals with the generation of an acyclic digraph, which will be used in the direct and in the contraction method. In Subsection 3.2.2 we treat the case of cycle structures.

3.2.1 Acyclic Network Structure

Suppose that the following input data for the generation of acyclic digraph $\dot{G} =< V, E >$ are given:

V	set of nodes
$P_{n+1}^{min}, P_{n+1}^{max}$	minimum and maximum number of sinks in $\dot{G}$
S_0^{min}, S_0^{max}	minimum and maximum numner of sources in $\dot{G}$
P^{max}	maximum number of non-redundant arcs entering node $j \in V$
S^{max}	maximum number of non-redundant arcs leaving node $i \in V$
RT	restrictiveness of Thesen for acyclic weak components
ρ	degree of redundancy in $\dot{G}$, that is, the percentage of redundant arcs in E relative to m_{red}^{max}

Algorithm A4. Generation of an acyclic digraph without redundancy

Set $V := \{1,...,n\}$ and $E := \varnothing$. Initialize adjacency matrix $\boldsymbol{A} := \boldsymbol{O}$ and reachability matrix $\boldsymbol{R} := \boldsymbol{I}$.

(1) *Generation of sources and sinks*

Determine a random number $r := rand\{S_0^{min},...,S_0^{max}\}$ of sources and a random number $s := rand\{P_{n+1}^{min},...,P_{n+1}^{max}\}$ of sinks.

Let $1,...,r$ be the sources of $\dot{G}$: $P_j^{max} := 0 \; \forall j \in R := \{1,...,r\}$.

Let $n-s+1,...,n$ be the sinks of $\dot{G}$: $S_i^{max} := 0 \; \forall i \in S := \{n-s+1,...,n\}$.

$P_j^{max} := P^{max}$, $S_j^{max} := S^{max} \; \forall j \in \{r+1,...,n-s\}$.

(2) *Generation of direct predecessors*

$V_p := V \setminus R$.

WHILE $V_p \neq \varnothing$ DO

Select randomly a node $j \in V_p$.

$V_p := V_p \setminus \{j\}$.

Determine set P_j of possible predecessors of node *j*:

$P_j := \{i \in V \mid r_{ij} + \rho_{ij} = 0, r_{ji} = 0, \delta^+(i) = 0\}$.

IF $P_j = \varnothing$ THEN

$P_j := \{i \in V \mid \rho_{ij} = 0, r_{ji} = 0, \delta^+(i) < S_i^{max}\}$.

END (* IF *).

Select randomly a node $i \in P_j$.

Insert arc $< i, j >$ in $\dot{G}$: $E := E \cup \{< i, j >\}$.

Update sets $P_j, P, S, \bar{R}$, and R :

$P_j := P_j \setminus (\bar{R}(i) \cup R(i))$

$P(j) := P(j) \cup \{i\}$.

$S(i) := S(i) \cup \{j\}$.

$\bar{R}(l) := \bar{R}(l) \cup \{k\}\ \forall (k,l): k \in \bar{R}(i), l \in R(j)$.

$R(k) := R(k) \cup \{l\}\ \forall (k,l): k \in \bar{R}(i), l \in R(j)$.

Update matrices $\boldsymbol{A}$, $\boldsymbol{R}$, and $\boldsymbol{R}^2$ (cf. Figure 1):

$\boldsymbol{A}$: $a_{ij} := 1$.

$\boldsymbol{R}$: $r_{kl} := 1\ \forall (k,l): k \in \bar{R}(i), l \in R(j)$.

$\boldsymbol{R}^2$: $r_{kl}^{(2)} := |R(k) \cap \bar{R}(l)|\ \forall (k,l): k \in \bar{R}(i), l \in R(j)$.

END (* WHILE *).

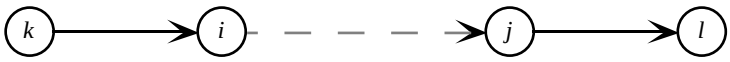


Fig. 1. Insertion of arc $< i, j >$

(3) *Generation of direct successors*

$V_s := V \setminus R$.

WHILE $V_s \neq \varnothing$ DO

Select randomly a node $i \in V_s$.

$V_s := V_s \setminus \{i\}$.

Determine set S_i of possible successors of node i:

$S_i := \left\{ j \in V \middle| r_{ij} + \rho_{ij} = 0, r_{ji} = 0, \delta^-(j) < P_j^{max} \right\}$.

Select randomly a node $j \in S_i$.

Insert arc $< i, j >$ in $\dot{G}$: $E := E \cup \{< i, j >\}$.

Update sets $S_i, P, S, \bar{R}$, and R :

$S_i := S_i \setminus (\bar{R}(j) \cup R(j))$

$P(j) := P(j) \cup \{i\}$.

$S(i) := S(i) \cup \{j\}$.

$\bar{R}(l) := \bar{R}(l) \cup \{k\}\ \forall (k,l): k \in \bar{R}(i), l \in R(j)$.

$R(k) := R(k) \cup \{l\}\ \forall (k,l): k \in \bar{R}(i), l \in R(j)$.

Update matrices $\boldsymbol{A}$, $\boldsymbol{R}$, $\boldsymbol{R}^2$:

$\boldsymbol{A}$: $a_{ij} := 1$.

$\boldsymbol{R}$: $r_{kl} := 1\ \forall (k,l): k \in \bar{R}(i), l \in R(j)$.

$\boldsymbol{R}^2$: $r_{kl}^{(2)} := |R(k) \cap \bar{R}(l)|\ \forall (k,l): k \in \bar{R}(i), l \in R(j)$.

END (* WHILE *).

(4) *Generation of additional arcs which do not generate redundancy*

Determine the estimator for the restrictiveness *rt* of $\dot{G}$:

$rt = \dfrac{2\sum_{i,j\in V} r_{ij} - 2n}{n(n-1)}$ (cf. 2.11; notice that the supersource 0 and the supersink $n+1$ have not been introduced).

IF $rt < RT$ THEN
Determine the set of nodes P whose indegree can be increased:
$P := \left\{ i \in V \middle| \delta^+(i) < S_i^{max} \right\}$.
WHILE $rt < RT$ DO
Select randomly a node $i \in P$.
Determine the set S_i of possible successors of node *i*:
$S_i := \left\{ j \in V \middle| r_{ij} + \rho_{ij} = 0, r_{ji} = 0, \delta^-(j) < P_j^{max} \right\}$.
IF $S_i = \varnothing$ THEN
$P := P \setminus \{i\}$
ELSE
Select randomly a node $j \in S_i$.
Insert arc $< i, j >$ in $\dot{G}$: $E := E \cup \{< i, j >\}$.
Update sets $P, P, S, \bar{R}$, and R :
IF $\delta^+(i) = S_i^{max}$ THEN
$P := P \setminus \{i\}$.
END (* IF *).
$P(j) := P(j) \cup \{i\}$.
$S(i) := S(i) \cup \{j\}$.
$\bar{R}(l) := \bar{R}(l) \cup \{k\}\ \forall (k,l): k \in \bar{R}(i), l \in R(j)$.
$R(k) := R(k) \cup \{l\}\ \forall (k,l): k \in \bar{R}(i), l \in R(j)$.
Update matrices $\boldsymbol{A}$, $\boldsymbol{R}$, $\boldsymbol{R}^2$, and restrictiveness estimator *rt*:
$\boldsymbol{A}$: $a_{ij} := 1$.
rt: $rt := rt + \dfrac{2}{n(n-1)} \left| \left\{ (k,l) \in V \times V \middle| r_{kl} = 0, k \in \bar{R}(i), l \in R(j) \right\} \right|$.
$\boldsymbol{R}$: $r_{kl} := 1\ \forall (k,l): k \in \bar{R}(i), l \in R(j)$.
$\boldsymbol{R}^2$: $r_{kl}^{(2)} := \left| R(k) \cap \bar{R}(l) \right|\ \forall (k,l): k \in \bar{R}(i), l \in R(j)$.
END (* IF *).
END (* WHILE *).
END (* IF *). ❑

Algorithm 5. Insertion of redundant arcs in a digraph without redundancy

Determine the number of redundant arcs to be inserted in $\dot{G}$: $m_{red} := \lfloor \rho m_{red}^{max} \rfloor$
(cf. Theorem 2).
$m_{nonRed} := |E|$.
$P := \left\{ i \in V \middle| \delta^+(i) < |V \setminus R| \right\} \setminus S$.
WHILE $|E| < m_{nonRed} + m_{red}$ DO
    Select randomly a node $i \in P$.
    IF i has been selected for the first time THEN
        Determine the set S_i of possible successors of node i:
        $S_i := \left\{ j \in V \setminus R \middle| r_{ij} = 1, a_{ij} = 0, r_{ji} = 0 \right\}$.
    END (* IF *).
    IF $S_i = \emptyset$ THEN
        $P := P \setminus \{i\}$.
    ELSE
        Select randomly a node $j \in S_i$.
        Insert arc $< i, j >$ in $\dot{G}$: $E := E \cup \{< i, j >\}$.
        Update sets P, S_i, P, and S ($\overline{R}$ and R remain unchanged):
        IF $\delta^+(i) = |V \setminus R|$ THEN
            $P := P \setminus \{i\}$.
        END (* IF *).
        $S_i := S_i \setminus \{j\}$.
        $P(j) := P(j) \cup \{i\}$.
        $S(i) := S(i) \cup \{j\}$.
        Update matrix $\boldsymbol{A}$ (matrices $\boldsymbol{R}$ and $\boldsymbol{R}^2$ remain unchanged):
        $\boldsymbol{A}$: $a_{ij} := 1$.
    END (* IF *).
END (* WHILE *). ❑

Remark 10.

The application of Algorithm A5 does not influence the restrictiveness of $\dot{G}$.

Remark 11.

The insertion of an arc $< i, j >$ in Algorithm A5 generates exactly one redundant arc, namely $< i, j >$. If we replaced $r_{ij} = 1$ by $r_{ij} + \rho_{ij} > 0$ for the determination of set S_i, we would not restrict ourselves to redundant arcs $< i, j >$ but could also insert redundancy-generating arcs $< i, j >$, which are not redundant. In this case, however, the insertion of arc $< i, j >$ would increase the restrictiveness of $\dot{G}$.

3.2.2 Cycle Structures

Suppose that the following input data for the generation of cycle structures in the acyclic digraph $\dot{G} =< V,E >$ are given:

$MTL^{min}, MTL^{max} \in [0,1]$	minimum and maximum percentage of maximum time lags, respectively
CS^{min}, CS^{max}	minimum and maximum number of cycle structures, respectively
n_c^{min}, n_c^{max}	minimum and maximum cardinal number of a cycle structure, respectively
$\delta \in [0,1]$	percentage of arcs employed for cycle structure densification

Algorithm 6. Generation of the structure of a cyclic network: Direct method

Determine randomly the number *t* of backward arcs corresponding to maximal time lags which will be inserted in $\dot{G}$: $t := rand\left\{\left\lceil |E| MTL^{min} \right\rceil, \ldots, \left\lfloor |E| MTL^{max} \right\rfloor\right\}$.

Determine randomly the number *CS* of cycle structures which are to be generated in $\dot{G}$: $CS := rand\left\{CS^{min}, \ldots, \min\left\{t, CS^{max}\right\}\right\}$.

Initialize the set of cycle structures $C := \varnothing$.

(1) *Creation of cycle structures*

Set $P := V$.

WHILE $\Gamma := \sum_{\substack{i \in V \\ r_{ii}^{(2)} > 1}} \frac{1}{r_{ii}^{(2)}} < CS$ DO (cf. Corollary 2)

Select randomly a node $i \in P$.

Determine the set T_i of nodes *j* for which a maximal time lag T_{ij}^{max} (a corresponding backward arc $< j,i >$, respectively) can be introduced:

$$T_i := \left\{ j \in V \setminus \{i\} \,\middle|\, \sum_{c \in C} r_{ic} r_{cj} = 0, n_c^{min} \le r_{ij}^{(2)} \le n_c^{max} \right\}$$ (cf. Theorem 3).

Select randomly a node $j \in T_i$.

Insert arc $< j,i >$ in $\dot{G}$: $E := E \cup \left\{< j,i >\right\}$.

$t := t - 1$.

Update set of cycle structures: $C := C \cup \left\{\dot{C}(i)\right\}$.

Update sets P, P, S, R, and $\bar{R}$:

$P := P \setminus (\bar{R}(j) \cap R(i))$

$P(i) := P(i) \cup \{j\}$.

$S(j) := S(j) \cup \{i\}$.

$R(h) := R(h) \cup \{g\}\ \forall (h,g): h \in \bar{R}(j), g \in R(i)$.

$\bar{R}(g) := \bar{R}(g) \cup \{h\}\ \forall (h,g): h \in \bar{R}(j), g \in R(i)$.

Update matrices $\boldsymbol{A}$, $\boldsymbol{R}$, and $\boldsymbol{R}^2$ (cf. Figure 2):

$\boldsymbol{A}$: $a_{ji} := 1$.

$\boldsymbol{R}$: $r_{hg} := 1\ \forall (h,g): h \in \bar{R}(j), g \in R(i)$.

$\boldsymbol{R}^2$: $r_{hg}^{(2)} := \left|\bar{R}(g) \cap R(h)\right|\ \forall (h,g): h \in \bar{R}(j), g \in R(i)$.

Let c be the contracted cycle structure of $\dot{C}(i) = \dot{C}(j)$.

Insert new column c and new row c in matrices $\boldsymbol{R}$ and $\boldsymbol{R}^2$:

$r_{hc} := 1\ \forall h \in \bar{R}(j)$.

$r_{cg} := 1\ \forall g \in R(i)$.

END (* WHILE *).

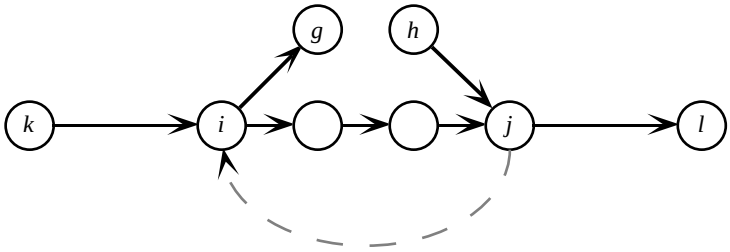


Fig. 2. Insertion of arc $< j, i >$

(2) *Extension of cycle structures*

Determine the number t_e of arcs to be used for the extension of cycle structures: $t_e := |C| + \lfloor (1-\delta)(t - |C|) \rfloor$.

Set $P := V$.

WHILE $t_e > 0$ DO

Select randomly a node $i \in P$.

Determine the set T_i of nodes j for which a maximal time lag T_{ij}^{max} (a corresponding backward arc $< j, i >$, respectively) can be introduced:

$$T_i := \left\{ j \in V \setminus \{i\} \,\middle|\, r_{ij} = 1, \sum_{c \in C} r_{ic} r_{cj} = 1, n_c^{min} \le r_{ij}^{(2)} \le n_c^{max} \right\}.$$

IF $T_i = \varnothing$ THEN

$P := P \setminus \{i\}$.

ELSE

Select randomly a node $j \in T_i$.

Insert arc $< j,i >$ in $\dot{G}$: $E := E \cup \{< j,i >\}$.
$t := t - 1$.
$t_e := t_e - 1$.
Update sets P, S, R, and $\bar{R}$:
$P(i) := P(i) \cup \{j\}$.
$S(j) := S(j) \cup \{i\}$.
$R(h) := R(h) \cup \{g\}\ \forall (h,g): h \in \bar{R}(j), g \in R(i)$.
$\bar{R}(g) := \bar{R}(g) \cup \{h\}\ \forall (h,g): h \in \bar{R}(j), g \in R(i)$.
Update matrices $\boldsymbol{A}$, $\boldsymbol{R}$, and $\boldsymbol{R}^2$:
$\boldsymbol{A}$: $a_{ji} := 1$.
$\boldsymbol{R}$: $r_{hg} := 1\ \forall (h,g): h \in \bar{R}(j), g \in R(i)$,
$\boldsymbol{R}^2$: $r_{hg}^{(2)} := \left|\bar{R}(g) \cap R(h)\right|\ \forall (h,g): h \in \bar{R}(j), g \in R(i)$.
Let c be the contracted cycle structure $\dot{C}$ with $\{i,j\} \cap V(\dot{C}) \neq \emptyset$.
$r_{hc} := 1\ \forall h \in \bar{R}(j)$.
$r_{cg} := 1\ \forall g \in R(i)$.
END (* IF *).
END (* WHILE *).

(3) *Densification of cycle structures*

Set $P := V$.
WHILE $t > 0$ DO
Select randomly a node $i \in P$.
IF i has been selected for the first time THEN
Determine the set T_i of nodes j for which a maximal time lag T_{ij}^{max} (a corresponding backward arc $< j,i >$, respectively) can be introduced:
$T_i := \left\{ j \in V \setminus \{i\} \middle| a_{ji} = 0, r_{ij} = 1, r_{ji} = 1 \right\}$.
END (* IF *).
IF $T_i = \emptyset$ THEN
$P := P \setminus \{i\}$.
ELSE
Select randomly a node $j \in T_i$.
Insert arc $< j,i >$ in $\dot{G}$: $E := E \cup \{< j,i >\}$.
$t := t - 1$.

Update sets T_i, P, and S (R and $\bar{R}$ remain unchanged):

$T_i := T_i \setminus \{j\}$.

$P(i) := P(i) \cup \{j\}$.

$S(j) := S(j) \cup \{i\}$.

Update matrix $\boldsymbol{A}$ ($\boldsymbol{R}$ and $\boldsymbol{R}^2$ remain unchanged):

$\boldsymbol{A}$: $a_{ji} := 1$.

END (* IF *).

END (* WHILE *). ❑

Let $\dot{G} = < V, E >$ be a digraph. The following algorithm introduces a supersource 0 and a supersink $n+1$, thus transforming $\dot{G}$ into a weakly connected digraph.

Algorithm A7. Addition of supersource 0 and supersink n+1

$V := V \cup \{0, n+1\}$.

$E := E \cup \{< 0, j > | j \in R\} \cup \{< i, n+1 > | i \in S\}$.

$a_{0,j} := 1 \; \forall j \in R$, $a_{i,n+1} := 1 \; \forall i \in S$.

$P(j) := P(j) \cup \{0\} \; \forall j \in R$, $S(i) := S(i) \cup \{n+1\} \; \forall i \in S$

$r_{0,j} := 1 \; \forall j \in V$, $r_{i,n+1} := 1 \; \forall i \in V$.

$R(0) := V, \bar{R}(n+1) := V$. ❑

The direct method for the generation of cyclic networks consists of the application of Algorithms A5, A6, and A7. Since cycle structures are constructed by the subsequent insertion of backward arcs in the acyclic digraph, it may happen that a feasible number $CS \leq \left\lfloor \frac{n}{n_c^{min}} \right\rfloor$ of cycle structures cannot be generated.

In the following, we develop another algorithm for network structure generation. The contraction method first constructs cycle structures which are then inserted in an acyclic network. Thus, any feasible number of cycle structures can be generated. On the other hand, the control of the restrictiveness must be limited to the generation of the acyclic skeleton of the isolated cycle structures in Step 1 and the construction of the aggregated network (including nodes corresponding to contracted cycle strucutures) in Step 4.

Algorithm A8. Generation of the structure of a cyclic network: Contraction method

Determine randomly the number t of backward arcs corresponding to maximal time lags which will be inserted in $\dot{G}$: $t \in \left\{\left\lceil |E| MTL^{min} \right\rceil, ..., \left\lfloor |E| MTL^{max} \right\rfloor\right\}$.
Determine randomly the number CS of cycle structures which are to be generated in $\dot{G}$: $CS \in \left\{CS^{min}, ..., \min\left\{t, CS^{max}\right\}\right\}$.

(1) *Generation of CS acyclic digraphs*

Determine a random partitioning $X := \left\{V_\nu | \nu = 1, ..., CS\right\}$ of a subset V' of the set V of nodes ($|V'| \in \left\{n_c^{min} CS, ..., n_c^{max} CS\right\}$) such that $|X| = CS$ and $\left|V_\nu\right| \in \left\{n_c^{min}, ..., n_c^{max}\right\}$ $\forall \nu = 1, ..., CS$.
Construct CS acyclic digraphs $\dot{G}_\nu =< V_\nu, E_\nu >$ by performing Steps 1, 2, and 3 of Algorithm A4. Let R_ν be the set of sources and let S_ν be the set of sinks of digraph $\dot{G}_\nu$.

(2) *Transformation of the CS acyclic digraphs into strongly connected digraphs*

Compute the number t^{min} of arcs required for the transformation of any digraph $\dot{G}_\nu =< V_\nu, E_\nu >$ to a strongly connected digraph $\dot{G}'_\nu =< V_\nu, E'_\nu >$ ($\nu = 1, ..., CS$):
$t^{min} := \sum_{\nu=1}^{CS} \max\left\{|R_\nu|, |S_\nu|\right\}$.
$t := \max\left\{t, t^{min}\right\}$. Determine a random vector $\boldsymbol{t} := (t_1, t_2, ..., t_{CS})$ with $t_\nu \geq \max\left\{|R_\nu|, |S_\nu|\right\}$ $\forall \nu = 1, ..., \text{CS}$ and $\sum_{\nu=1}^{CS} t_\nu = t$.

FOR $\nu = 1, ..., CS$ DO
$t_{\nu,d} := t_\nu - \max\left\{|R_\nu|, |S_\nu|\right\}$
Transform $\dot{G}_\nu =< V_\nu, E_\nu >$ to a strongly connected digraph $\dot{G}'_\nu =< V_\nu, E'_\nu >$ by applying Algorithm A9.
Perform Step 4 of Algorithm A4 and Algorithm A5.
Insert $t_{\nu,d}$ further arcs by densifying $\dot{G}'_\nu$ using Step 3 of Algorithm A6. Instead of V, set P has to be initialized with $P := V_\nu$ and the formula for the determination of set T_i has to be replaced by $T_i := \left\{j \in V_\nu \middle| a_{ji} = 0\right\}$.
END (* FOR *).

(3) *Contraction of all cycle structures*

Set $C := \bigcup_{\nu=1,\ldots,CS} \dot{G}'_\nu$. Replace the nodes of all cycle structures $\dot{G}'_\nu$ by the corresponding contracted cycle structure c_ν: $V := V \setminus \bigcup_{\nu=1,\ldots,CS} V(\dot{G}'\nu) \cup \bigcup_{\nu=1,\ldots,CS} \{c_\nu\}$.

(4) *Construction of an acyclic multidigraph*

Construct an acyclic digraph $\dot{G} = < V, E >$ based on the (new) node set V using algorithms A4 and A5. Since the contracted cycle structures actually consist of several nodes, an arc $< i, j >$ being incident with a node c_ν can be in parallel up to $\left|V(\dot{G}'_\nu)\right|$ times.

(5) *Expansion of cycle structures*

FOR $\nu = 1,\ldots,CS$ DO
  Set $\overline{E}_\nu := \{< i, j > \in E \mid i = c_\nu \vee j = c_\nu\}$.
  Expand cycle structure $\dot{G}'_\nu$ by replacing node c_ν in the node set V of $\dot{G}$ by $V(\dot{G}'_\nu)$.
  Update E: $E := E \setminus \overline{E}_\nu \cup E'_\nu$.
  Determine a random assignment of arcs $< i, j > \in \overline{E}_\nu$ to nodes $i \in V(\dot{G}'_\nu)$ or $j \in V(\dot{G}'_\nu)$, respectively:
  WHILE $\overline{E}_\nu \neq \emptyset$ DO
    Select an arc $< i, j > \in \overline{E}_\nu$.
    $\overline{E}_\nu := \overline{E}_\nu \setminus \{< i, j >\}$.
    IF $i = c_\nu$ THEN
      Select randomly a node $k \in V(\dot{G}'_\nu)$.
      Insert arc $< k, j >$ in $\dot{G}$: $E := E \cup \{< k, j >\}$.
      $a_{kj} := 1$.
    ELSE
      Select randomly a node $l \in V(\dot{G}'_\nu)$.
      Insert arc $< i, l >$ in $\dot{G}$: $E := E \cup \{< i, l >\}$.
      $a_{il} := 1$.
    END (* IF *).
  END (* WHILE *).
END (* FOR *).

(6) *Addition of supersource 0 and supersink n+1*

Add supersource 0 and supersink n+1 applying Algorithm A7. ❑

The following algorithm which is used in Step 2 of the contraction method transforms a digraph into a strong component inserting a minimal number of additional arcs.

Algorithm A9. Generation of a strong component $\dot{G}'=<V,E'>$ based on a given subdigraph $\dot{G}=<V,E>$

(1) *Generation of an acyclic weak component*

$\dot{G}':=\dot{G}$.
Compute the set C of all cycle structures of $\dot{G}'$ with an algorithm which can be found in Even (1979): $C:=\left\{\dot{C}(i)|i\in V\right\}$.
Contract all cycle structures $\dot{C}(i)\in C$.
Let R be the set of sources of $\dot{G}'$ and S be the set of sinks of $\dot{G}'$.
Let SC and SG be empty stacks.
Let $\left\{\dot{G}_1,\dot{G}_2,...,\dot{G}_{WC}\right\}$ be the set of weak components of $\dot{G}'$.
FOR $\nu=1,...,WC-1$ DO
    Let s be a sink of $\dot{G}_\nu$ and let r be a source of $\dot{G}_{\nu+1}$.
    Insert arc $<s,r>$.
    $R:=R\setminus\{r\},S:=S\setminus\{s\}$.
END (* FOR *).

(2) *Elimination of sources and sinks*

WHILE $|S|>0$ and $R\cap S=\varnothing$ DO
    IF $|R|=1$
        Select an arbitrary sink $s\in S$ and the unique source $r\in R$.
        Insert arc $<s,r>$.
        $R:=R\setminus\{r\}\cup\left\{\dot{C}(r)\right\},S:=S\setminus\{s\}$.
        IF $|S|=0$
            $S:=S\cup\left\{\dot{C}(r)\right\}$.
        END (*IF*).
    ELSIF $|S|=1$
        Select the unique sink $s\in S$ and an arbitrary source $r\in R$.
        Insert arc $<s,r>$.
        $R:=R\setminus\{r\},S:=S\setminus\{s\}\cup\left\{\dot{C}(r)\right\}$.
    ELSE
        Select a sink $s\in S$ with $|\bar{R}(s)\cap R|\geq 2$.
        Determine a source $r\in R$ with $s\in R(r)$ and $|R(r)\cap S|\geq 2$.
        Insert arc $<s,r>$.
        $R:=R\setminus\{r\},S:=S\setminus\{s\}$.
    END (* IF *).

$SC := SC \cup C(r)$. $SG := SG \cup \{\dot{G}'\}$.

Contract cycle structure $\dot{C}(r)$.

END (* WHILE *).

(3) *Expansion of the cycle structures*

WHILE $SC \neq \emptyset$ DO

$\dot{C}(r)$:= Head(SC).

$\dot{G}''$:= Head(SG).

Expand $\dot{C}(r)$ in $\dot{G}'$ with respect to $\dot{G}''$ in analogy to step 5 of Algorithm A8.

$SC := SC \setminus \{\dot{C}(r)\}$.

$SG := SG \setminus \{\dot{G}''\}$.

END (* WHILE *). ❑

Remark 12.

Applying Algorithm A9 in the contraction method, the algorithm for the identification of cycle structures can be skipped since $\dot{G}$ is acyclic.

Theorem 9.

(a) Algorithm A9 is correct (that is, it generates a strong component $\dot{G}'$ with subdigraph $\dot{G}$ in a finite number of steps).

(b) Among all strong components with subdigraph $\dot{G}$, the strongly connected digraph $\dot{G}'$ generated contains the minimum number of arcs.

Proof.

(a) The contraction of the cycle structures in Step 1 yields an acyclic digraph $\dot{G}'$ consisting of WC weak components, each having at least one source and one sink. The insertion of $WC-1$ arcs which weakly connect subdigraphs $\dot{G}_\nu$ and $\dot{G}_{\nu+1}$ ($\nu = 1,\ldots,WC-1$) obviously makes digraph $\dot{G}'$ a weak component. Like any acyclic digraph, $\dot{G}'$ has at least one source and one sink.

If $R \cap S \neq \emptyset$, $\dot{G}'$ consists of only one node and we skip to Step 3.

Since $s \in R(s)$ for each of the three cases, we generate a new cycle structure in every pass of Step 2 which is contracted in $\dot{G}'$, that is, each digraph $\dot{G}'$ obtained at the end of Step 2 is acyclic and weakly connected, too. Due to Theorem 5, there can always be selected a source $r \in R$ and a sink $s \in S$ satisfying the conditions specified in Step 2.

At any pass of Step 2, the number of nodes is reduced by at least one. The algorithm passes to Step 3, if there remains only one sink which represents a source at the same time. This is, due to the weak connectivity of all intermediate digraphs $\dot{G}'$, true exactly if there remains only one node. Since V is finite, this will be achieved after a finite number of passes of Step 2.
The successive expansion of contracted cycle structures in Step 3 finally yields a strong component which constitutes a subdigraph of the underlying digraph $\dot{G}$.

(b) The contraction of all cycle structures of digraph $\dot{G}$ leads to an acyclic digraph $\dot{G}'$ with set of sources R and set of sinks S. The number of arcs required for the transformation of $\dot{G}'$ into a strong component will be the same as the number of arcs required for the transformation of $\dot{G}$ into a strong component.
The insertion of an arc $< i, j > (i, j \in V)$ can at most eliminate one source and one sink. Obviously, the node set of a strong component does not contain sources or sinks. Therefore, the minimum number of arcs which are required to make $\dot{G}$ strongly connected is $\max\{|R|, |S|\}$.
In Step 1, in each insertion of an arc $< s, r >$ exactly one source and one sink are eliminated without creating a new cycle structure in $\dot{G}'$ ($s \notin R(r)$, since r and s are nodes which initially belong to two different weak components of $\dot{G}'$).
Considering Step 2, we have to distinguish between four cases.

(i) $|R| = 1, |S| \geq 2$

By the insertion of $< s, r >$ and the subsequent contraction of $\dot{C}(r)$ to node c, we eliminate source r and sink s, obtaining an additional source c.

(ii) $|R| \geq 2, |S| = 1$

By the insertion of $< s, r >$ and the subsequent contraction of $\dot{C}(r)$ to node c, we eliminate source r and sink s, obtaining an additional sink c.

(iii) $|R| \geq 2, |S| \geq 2$

By Theorem 6, the insertion of $< s, r >$ and the subsequent contraction of $\dot{C}(r)$ to node c reduces both the number of sources and the number of sinks by one.

(iv) $|R| = 1, |S| = 1$

By the insertion of $< s, r >$ and the subsequent contraction of $\dot{C}(r)$ to node c, we eliminate source r and sink s, obtaining an additional source c which constitutes a sink at the same time, that is, $|R| = 1, |S| = 1$ and $R \cap S \neq \emptyset$. $\dot{G}'$ consists of the single node c.

All in all, the algorithm inserts

$$\underbrace{WC-1}_{\text{Step 1}}+\underbrace{\underbrace{\big\||R|-|S|\big\|}_{\text{Case 1, 2}}+\underbrace{\min\{|R|,|S|\}-(WC-1)-1}_{\text{Case 3}}+\underbrace{1}_{\text{Case4}}}_{\text{Step 2}}=\max\{|R|,|S|\}$$

arcs in $\vec{G}$, which represents, as seen above, the minimum number required to achieve the strong connectivity of $\vec{G}'$. ❑

Remark 13. Time complexity of Algorithm A9

The time complexity of Step 1 of Algorithm A9 is $O(|E|)$, since $O(|E|)$ represents the time required for the identification of cycle structures.
Let R be the set of sources and S be the set of sinks in the acyclic digraph $\vec{G}'$ obtained at the end of Step 1. Then, the time complexity of Step 2 is $O(\min\{|R|,|S|\}(|R|+|S|))$.
The time complexity of Step 3 is $O(|E'|)=O(|E|+\max\{|R|,|S|\})$, since the expansion of cycle structure $\vec{C}(r)$ can be done in $O(|E(\vec{C}(r))|)$ and $\bigcup_{\vec{C}(r)\in C} E(\vec{C}(r))\subseteq E'$.
Hence, the time complexity of Algorithm A9 is $O(|E|+\min\{|R|,|S|\}(|R|+|S|))$.

3.3 Activity Durations and Arc weights

Let $\vec{G}=<V,E>$ be the weakly connected digraph generated in Subsection 3.2. (Forward) arcs generated in Subsection 3.2.1 belong to minimal time lags, (backward) arcs generated in Subsection 3.2.2 belong to maximal time lags. For the representation of precedence and time constraints by the project network $\vec{N}=<V,E;b>$ we have to determine a weight b_{ij} for any arc $<i,j>\in E$.
In this subsection we generate activity durations D_{jm} $(j\in V, m\in M_j)$. Based on these durations, minimal and maximal time lags (arc weights b_{ij}) are calculated for $<i,j>\in E$.

The following input data have to be specified by the user of ProGen/max:

D^{min}, D^{max}	integer-valued minimal and maximal duration of an activity
$\varepsilon_d\in[0,\infty)$	maximal relative deviation of minimal time lags from activity durations
$CST\in[0,1]$	cycle structure tightness
$SF\in[0,\infty)$	slack factor
$PDT\in[0,1]$	project duration tightness

Activity durations D_{jm} $(j\in V\setminus\{0,n+1\}, m\in M_j)$ are generated randomly out of the set $\{D^{min},\dots,D^{max}\}$: $D_{jm}:=rand\{D^{min},\dots,D^{max}\}$.

In contrast to the problem generator of Kolisch et al. (1995), the minimal time lags T_{ijm}^{min} between two activities i, j may be different from the duration D_{im} of activity *i*. This way, overlappings and waiting times between activities can be modeled. We introduce an index $\varepsilon_d \in [0,\infty)$ such that the minimal time lag T_{ijm}^{min} between the start of activity *i* and the start of activity *j* depending on the execution mode $m \in M_i$ of activity *i* lies in the interval $[(1-\varepsilon_d)D_{im},(1+\varepsilon_d)D_{im}]$. As mentioned in Section 2, $T_{ijm}^{min} < 0$ can be viewed as a positive maximal time lag $T_{jim}^{max} := -T_{ijm}^{min} > 0$ between activities *j* and *i*.

The maximal time lag T_{ij}^{max} between the start of activity *i* and the start of activity *j* is determined randomly in interval (cf. Figure 3)

(3.1) $\left[b(i,j)-(b(i,j)-a(i,j))CST^2, [b(i,j)-2(b(i,j)-a(i,j))CST+(b(i,j)-a(i,j))CST^2](1+SF)\right].$

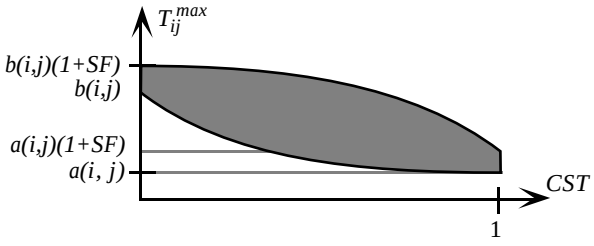


Fig. 3. Interval for maximal time lags depending on the cycle structure tightness *CST*

$a(i,j)$ represents the minimum time lag between the start of activity *i* and the start of activity *j* induced by the precedence constraints. $b(i,j)$ corresponds to a maximal time lag which can always be met (observing precedence and resource constraints).

Let $\dot{N}' = <V, E'; b'>$ be the weakly connected acyclic network with $b'_{kl} := \max_{m \in M_k} \left\{T_{klm}^{min}\right\}$ ($<k,l> \in E'$), where E' is the subset of E which contains all arcs corresponding to minimal time lags. $S'(i)$ denotes the set of direct successors of node *i* in $\dot{N}'$. $a(i,j)$ and $b(i,j)$ can be computed as follows:

(3.2) $a(i,j) := L_{\dot{N}'}^{r}(i,j)$:= length of a longest (directed) path from *i* to *j* in $\dot{N}'$.

(3.3) $$b(i,j) := \sum_{k \in R(i) \cap \bar{R}(j) \setminus \{j\}} \max_{m \in M_k} \max \left\{ D_{km}, \max_{l \in S'(k) \cap \bar{R}(j)} T_{klm}^{min} \right\}.$$

Note that $R(i)$ and $\bar{R}(j)$ denote the corresponding sets in (the cyclic network) $\dot{N}$.

If the problem instances to be generated are of type MRLP / max or MRIP / max, we have to determine an appropriate value for the project duration *T*. A lower bound T^{min} on the project duration is given by $T^{min} := L_{\dot{N}'}^{r}(0, n+1)$.

For each cycle structure $\dot{C} \in C$ in $\dot{N}$ we determine earliest start times EST_j and latest finish times LFT_j of activities $j \in V(\dot{C})$. An upper bound T^{max} on the project duration is then given by $T^{max} := \sum_{C \in C} \max_{j \in V(C)} LFT_j + \sum_{j \in V \setminus \bigcup_{C \in C} V(C)} \max_{m \in M_j} T_{ijm}^{min}$.

The project duration T is determined as follows: $T := T^{min} + \text{int}(PDT(T^{max} - T^{min}))$.

4. Resource Demand and Availability Generation

The third kind of restrictions besides precedence and time constraints given by the project network are limitations due to scarce resources. The relationship between resource demand and resource availability strongly influences the set of feasible subsets (cf. Mingozzi at al. 1994). A feasible subset *F* is defined to be a subset of the set of activities such that all activities of *F* can be executed at the same time taking precedence, time, and resource constraints into account (evidently, *F* depends on the modes in which the activities are performed).
Numerous resource characteristics for resource-constrained scheduling problems can be found in literature, for example in Kurtulus and Davis (1975), Patterson (1976), Davis (1982), Kurtulus and Narula (1985), and Kolisch et al. (1995).
Generalizing and normalizing measures which have been used before, Kolisch et al. (1995) developed a new set of control parameters which have a strong impact on the hardness of problem instances. ProGen/max employs the same set of resource measures for the problem generation. In the following, we briefly sketch the generation of resource requirements and resource availability proposed by Kolisch et al. (1995).

4.1 Resource demand

The processing of an activity consumes or uses a certain amount of one or several nonrenewable, renewable or doubly-constrained resources. After the generation of a certain number of resources, the generation of resource consumption and resource usage is performed in two steps: First, for any given activity-mode combination (j,m) with $j \in V, m \in M_j$ we have to determine a set R_{jm} of resources required for the processing of activity *j* in mode *m*. Then, for all resources $i \in R_{jm}$ we fix the (integer-valued) number of units which will be consumed or used for the processing of activity *j* in mode *m* $(j \in V, m \in M_j)$.

The generalized resource factor *RF* for multi-mode problems which has been introduced by Kolisch et al. (1995) indicates the mean percentage of resources which are affected by the execution of an activity:

(4.1) $RF_\rho := \frac{1}{|V|-2} \frac{1}{\left|R^\rho \setminus R^\nu\right|} \sum_{j \in V \setminus \{0,n+1\}} \frac{1}{\left|M_j\right|} \sum_{m \in M_j} \sum_{i \in R^\rho \setminus R^\nu} \delta(r_{ijm}^\rho)$ for renewable resources,

$RF_\nu := \frac{1}{|V|-2} \frac{1}{\left|R^\nu \setminus R^\rho\right|} \sum_{j \in V \setminus \{0,n+1\}} \frac{1}{\left|M_j\right|} \sum_{m \in M_j} \sum_{i \in R^\nu \setminus R^\rho} \delta(r_{ijm}^\nu)$ for nonrenewable resources,

$RF_\delta := \frac{1}{|V|-2} \frac{1}{\left|R^\rho \cap R^\nu\right|} \sum_{j \in V \setminus \{0,n+1\}} \frac{1}{\left|M_j\right|} \sum_{m \in M_j} \sum_{i \in R^\rho \cap R^\nu} \delta(r_{ijm}^\rho)$ for doubly-constrained resources

with $\delta(x) := \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases}$.

ProGen / max employs the following input data for the resource demand generation:

$Q_\tau^{min}, Q_\tau^{max}$ $(\tau \in \{\rho, \nu, \delta\})$	minimal and maximal number of renewable, nonrenewable, and doubly-constrained resources used by an activity ("request")
$RF_\tau^{min}, RF_\tau^{max}$ $(\tau \in \{\rho, \nu, \delta\})$	minimal and maximal resource factor of renewable, nonrenewable, and doubly-constrained resources
$U_\tau^{min}, U_\tau^{max}$ $(\tau \in \{\rho, \nu\})$	minimal and maximal number of units of renewable, nonrenewable, and doubly-constrained resources required for the processing of an activity ("level of demand")

For the first step, the algorithmic generation of the three-dimensional resource-activity-mode-array $RQ := \left(\delta(r_{ijm}^\tau)\right)_{i \in R^\rho \cup R^\nu, j \in V, m \in M_j}$, based on $Q_\tau^{min}, Q_\tau^{max}$ and $RF_\tau^{min}, RF_\tau^{max}$ $(\tau \in \{\rho, \nu, \delta\})$ we refer to Kolisch et al. (1995).

In the second step, we assign a demand level to any triplet (i, j, m) with $\delta(r_{ijm}^\tau) = 1$: $r_{ijm}^\tau := rand\left\{U_\tau^{min}, \ldots, U_\tau^{max}\right\}$ $(i \in R^\rho \cup R^\nu, j \in V, m \in M_j, \delta(r_{ijm}^\tau) = 1)$. In contrast to ProGen, for a given activity *j* and a given resource *i*, the resource requirements r_{ijm}^τ can vary with modes $m \in M_j$, if the respective option (mode-varying resource demand levels) has been selected. The generation of inefficient modes in the second step (that is, modes $\overline{m} \in M_j$: $(\exists\, m \in M_j : D_{jm} \le D_{j\overline{m}}, r_{ijm}^\rho \le r_{ij\overline{m}}^\rho \;\forall i \in R^\rho, r_{ijm}^\nu \le r_{ij\overline{m}}^\nu \;\forall i \in R^\nu)$) may necessitate several passes of the algorithm.

4.2 Resource Availability

Let $R_{i,\rho}^{min} := \max_{j \in V} \min_{m \in M_j} r_{ijm}^{\rho}$ be the minimal availability of renewable or doubly-constrained resource $i \in R^{\rho}$ required to perform all activities of the project and let $R_{i,\nu}^{min} := \sum_{j \in V} \min_{m \in M_j} r_{ijm}^{\nu}$ be the minimal availability of nonrenewable or doubly-constrained resource $i \in R^{\nu}$ required to perform all activities of the project.

In case of resources $i \in R^{\nu}$, an upper bound on the maximal availability required to perform all activities of the project can be calculated as follows: $R_{i,\nu}^{max} := \sum_{j \in V} \max_{m \in M_j} r_{ijm}^{\nu}$. For resources $i \in R^{\rho}$ we perform a resource-unconstrained temporal analysis in network $N'(i) := < V, E', b'(i) >$, where E' is the subset of the project network arc set E including all (forward) arcs which correspond to minimal time lags. Arcs $< j,l > \in E'$ are weighted with $b'_{jl}(i) := \min \left\{ T_{jlm_{ij}^*}^{min} \middle| m_{ij}^* = \arg\max_{m \in M_j} r_{ijm}^{\rho} \right\}$. Let $V_i(t) := \left\{ j \in V \middle| t - D_{jm_{ij}^*} < EST_j \le t \right\}$ be the set of activities in progress at time t based on earliest start times EST_j ($j \in V$) determined by the temporal analysis in $N'(i)$. Then, $R_{i,\rho}^{max} := \max_{t=0,\ldots,\overline{T}-1} \sum_{j \in V(t)} r_{ijm_{ij}^*}^{\rho}$ represents an upper bound on the maximal availability of a resource $i \in R^{\rho}$ required to perform all activities of the project.

The [0,1]-normalized resource strength $RS_{i,\tau}$ of resource i introduced by Kolisch et al. (1995) is defined as follows:

$$RS_{i,\tau} := \frac{R_i^{\tau} - R_{i,\tau}^{min}}{R_{i,\tau}^{max} - R_{i,\tau}^{min}} \quad (i \in R^{\tau}, \tau \in \{\rho, \nu\}). \tag{4.2}$$

The following input data is required for the generation of resource availability:

$RS_{\tau}^{min}, RS_{\tau}^{max}$ ($\tau \in \{\rho, \nu\}$)	minimal and maximal resource strength of renewable, nonrenewable, and doubly-constrained resources.

The resource strength is randomly determined as follows: $RS_{i,\tau} := rand[RS_{\tau}^{min}, RS_{\tau}^{max}]$. Hence, in contrast to ProGen, the resource strength of resources belonging to the same type $\tau \in \{\rho, \nu\}$ may be different if $RS_{\tau}^{min} < RS_{\tau}^{max}$.

With $R_i^{\tau} := R_{i,\tau}^{min} + \text{int}(RS_{i,\tau}(R_{i,\tau}^{max} - R_{i,\tau}^{min}))$ $(i \in R^{\tau}, \tau \in \{\rho, \nu\})$ we obtain the availability of renewable, nonrenewable, and doubly-constrained resources.

5. Computational Results

5.1 Accuracy of Thesen's Restrictiveness Estimator *RT*

In contrast to the problem generator ProGen by Kolisch et al. (1995), ProGen/max measures the complexity of the acyclic network structure by the estimator *RT* for the restrictiveness of a network (cf. Section 2.3). Since the hardness of many combinatorial optimization problems depends heavily on the underlying network structure, the parameters controlling the network generation should be chosen with care. In order to assess the quality of estimator *RT*, we have generated 330 acyclic networks with 10 activities and *RT* values between 0.2 and 1.0. For each network the exact restrictiveness P has been determined by complete enumeration. Figure 4 plots the relationship between values of the complexity measures *RT* and *CNC* and the restrictiveness resulting for the 330 networks.

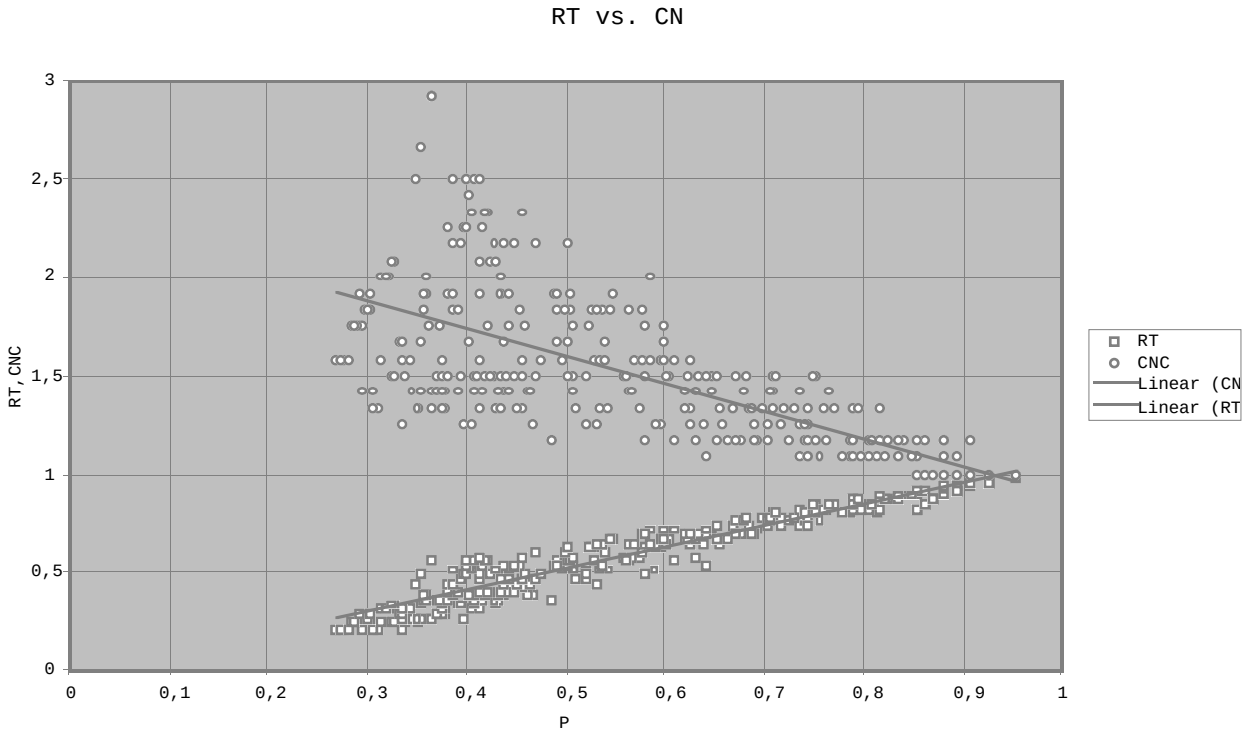


Fig. 4. *RT*(P) and *CNC*(P)

We have performed a linear regression relating *RT* and *CNC* to P. The results are as follows (r^2 denotes the squared correlation coefficient):

$$\mathrm{P} = 0.072 + 0.85RT \qquad (r^2 = 0.923)$$
$$\mathrm{P} = 1.092 - 0.346CNC \qquad (r^2 = 0.485)$$

The estimator *RT* accounts for more than 92% of the restrictiveness' variance. For small values, *RT* tends to overestimate P, whereas a large restrictiveness is generally underrated. These results coincide with the results obtained by Thesen (1977).

The behaviour of the *CNC* coefficient of network complexity explains the poor performance of this control parameter in predicting the computational effort of enumerative algorithms in project scheduling. The explanatory power of *CNC* in the linear regression model is less than 50%. Moreover, *CNC* is negatively correlated with P. This can be motivated as follows. The insertion of an additional arc always results in an increasing *CNC*, even if the arc is redundant. Redundant arcs, however, do affect neither the restrictiveness of the project network nor the hardness of corresponding project scheduling problems. This explains results obtained by Kolisch et al. (1995) which establishe a (slight) reduction in computation time with increasing *CNC*.

5.2 Hardness of RCPSP/max Instances

Problem generators should allow for a parameter-driven generation of easy and hard problem instances. In order to assess the impact of ProGen/max control parameters on the hardness of resource-constrained project scheduling problems we have generated 810 instances of the single mode problem RCPSP/max with the direct method of ProGen/max. We used a full factorial design for four parameters, which have proven to be closely related to the computational effort required for the (exact) solution of RCPSP instances (cf. Kolisch et al. 1995 and de Reyck 1995). Table 2 provides the constant parameter values. Table 3 shows the variable parameter values of the full factorial design experiment.

R^{min}	R^{max}	S_0^{min}	S_0^{max}	P_{n+1}^{min}	P_{n+1}^{max}	P^{max}	S^{max}
5	5	1	5	1	5	3	3

ρ	MTL^{min}	MTL^{max}	CS^{min}	CS^{max}	n_c^{min}	n_c^{max}	δ
0.05	0.05	0.25	1	5	2	5	0.5

D^{min}	D^{max}	ε_d	CST	SF	Q^{min}	Q^{max}	U^{min}	U^{max}
5	15	2.0	0.5	0.0	1	5	1	3

Table 2. Constant parameter values

n	RT	RF	RS
10	0.25	0.5	0.0
15	0.5	0.75	0.25
20	0.75	1.0	0.5

Table 3. Variable parameter values

For every combination of *n*, *RT*, *RF*, and *RS* we have generated 10 problem instances. 524 of the 810 problem instances turned out to have a feasible solution. Each problem instance has been treated by the branch-and-bound algorithm of de Reyck & Herroelen (1996) which currently has to be considered as the most advanced procedure for RCPSP/max. We used a version of the algorithm which has been recoded in the object-oriented

language Smalltalk-80. First, we determined the computation time for the generation of an optimal (or best) solution and the computation time for the verification of optimality, where we imposed a time limit of 600 seconds. Second, the deviation of the first generated solution from the optimal (or best) solution and the deviation of the latter solution from the tightest of several time-based and resource-based lower bounds has been calculated.

97 out of the 524 solvable problem instances could not be solved to optimality (i.e., verification of optimality) within the time limit. For all 524 solvable problem instances, however, a feasible solution could be determined. A first feasible solution could be determined in 2.326 s on the average. This first solution had a mean deviation of 3.37% from the best solution found, whereas the deviation of the best solution found from the tightest lower bound has been 6.99%. The best solution has been determined within 5.543 s on the average. The verification of optimality, however, required a much larger mean computation time of 92.314 s.

Table 4 shows the effect of increasing number n of activities on the computation times μ_{CPU}^{ver} and μ_{CPU}^{fd} for the verification and the determination of an optimal solution in seconds, respectively. $p_{unsolved}$ denotes the percentage of problem instances for which the best solution found could not be verified to be an optimum. $DevFS_{BS}$ represents the mean percentage deviation of the first generated solution from the best solution. $DevBS_{LB}$ denotes the mean percentage deviation of the best solution from the tightest lower bound. Since for 81.5% of the best solutions the optimality could be proven, $DevBS_{LB}$ can be interpreted as a tightness measure of lower bounds.

n	μ_{CPU}^{ver}	μ_{CPU}^{fd}	$p_{unsolved}$	$DevFS_{BS}$	$DevBS_{LB}$
10	16.005	3.924	1.09%	3.60%	7.32%
15	100.198	28.154	12.35%	3.77%	7.02%
20	164.022	45.517	22.47%	2.79%	6.61%

Table 4. Effects of number of activities n on problem hardness

Since RCPSP / max constitutes a combinatorial optimization problem which is known to be *NP*-hard in the strong sense, the increase in the number of activities may lead to computation times which cannot be bounded by a polynomial in n. The relatively small increase of μ_{CPU}^{ver} and μ_{CPU}^{fd} when n is enlarged from 15 to 20 is mainly due to the imposed time limit. Nevertheless, the implicit enumeration algorithm of de Reyck & Herroelen (1996) seems to perform the better the larger the problems are. The deviation of the first solution from the best solution decreases from 3.60% to 2.79% which can be partially explained by the increasing quality of the first solution and the lower bounds (note that $DevBS_{LB}$ decreases although the percentage $p_{unsolved}$ of not necessarily optimal solutions increases!).

The impact of varying the restrictiveness of the project network can be seen in Table 5.

$RT \in$	μ_{CPU}^{ver}	μ_{CPU}^{fd}	$p_{unsolved}$	$DevFS_{OPT}$	$DevOPT_{LB}$
[0,0.3]	183.958	53.458	26.76%	5.66%	10.31%
(0.3,0.6]	94.073	24.049	10.68%	3.55%	6.38%
(0.6,1]	16.318	4.773	1.14%	1.33%	5.02%

Table 5. Effects of the restrictiveness *RT* on problem hardness

The restrictiveness of the underlying project network turns out to have even a stronger impact on the hardness of RCPSP/max instances than the number *n* of project activities. Computation times increase heavily with growing parallelity of the project networks. Moreover, the quality of first solutions and lower bounds decreases for harder problems.

Table 6 depicts the impact of increasing the resource factor.

RF	μ_{CPU}^{ver}	μ_{CPU}^{fd}	$p_{unsolved}$	$DevFS_{OPT}$	$DevOPT_{LB}$
0.5	52.476	21.204	7.33%	2.26%	5.33%
0.75	92.162	19.197	10.98%	3.71%	7.61%
1.0	140.038	37.588	18.13%	4.35%	8.29%

Table 6. Effects of the resource factor *RF* on problem hardness

It is quite intuitive that larger resource factors result in an increase in resource conflicts. Each resource conflict generates several enumeration nodes in the enumeration tree of the de Reyck algorithm. Moreover, the poor quality of the first solution and lower bounds for high values of *RF* enlarges the gap between the (first) upper bound and the lower bounds.

The effects of varying the resource strength are summarized in Table 7.

RS	μ_{CPU}^{ver}	μ_{CPU}^{fd}	$p_{unsolved}$	$DevFS_{OPT}$	$DevOPT_{LB}$
0.0	106.148	41.784	11.36%	5.31%	10.20%
0.25	138.930	33.643	20.00%	4.24%	9.95%
0.5	53.108	13.880	5.98%	2.06%	3.68%

Table 7. Effects of the resource strength *RS* on problem hardness

Table 7 shows an interesting relationship between computation times and scarcity of (renewable) resources. In the interval [0.25,1], a decrease in resource availability makes the problems much more harder, since the number of resource conflicts increases heaviliy in the number of resource conflicts. The behaviour of μ_{CPU}^{ver}, μ_{CPU}^{fd}, and $p_{unsolved}$ for values of the resource strength between 0 and 0.25 seems to confirm the conjecture by Elmaghraby and Herroelen (1980) of a bell-shaped function $\mu_{CPU}^{ver}(RS)$ which had not been confirmed by Kolisch et al. (1995). Moreover, the suitability of *RS* (which has been consid-

ered in the past as the parameter with the strongest impact on computation times) for the distiction of hard and intractable problem instances is clearly outperformed by the restrictiveness estimator *RT* of the underlying project network.

Conclusions

We have developed a problem generator ProGen/max for the resource-constrained minimum project duration problem MRCPSP/max, the resource leveling problem MRLP/max, and the resource investment problem MRIP/max. The emphasis has been put on the efficient and parameter-driven generation of cyclic network structures, which occur if maximal time lags have to be taken into consideration.
Two different approaches to the generating cyclic networks have been proposed: the direct and the contraction method. The direct method focuses on the parameter-driven generation of the (entire) project network, whereas the contraction method emphasizes the structure of strong components (cycle structures) and the contracted network.
The impact of the selected control parameter configuration on the hardness of resource-constrained project scheduling problems has been evaluated by a full factorial design experiment. It turns out that the computational effort for the solution of problem instances heavily depends on both the degree of scarcity of resources and the restrictiveness of the underlying project network.

Appendix: Functional Description of ProGen/max

ProGen / max has been coded in the object-oriented programming language Smalltalk-80. The image file is portable to any platform supported by VisualWorks® Smalltalk (UNIX, Microsoft Windows, OS / 2, and Macintosh platforms). To start ProGen / max the image file has to be executed with the platform-dependent object engine which runs Smalltalk on the respective platform.

The control menue of ProGen / max is shown in Figure 4.

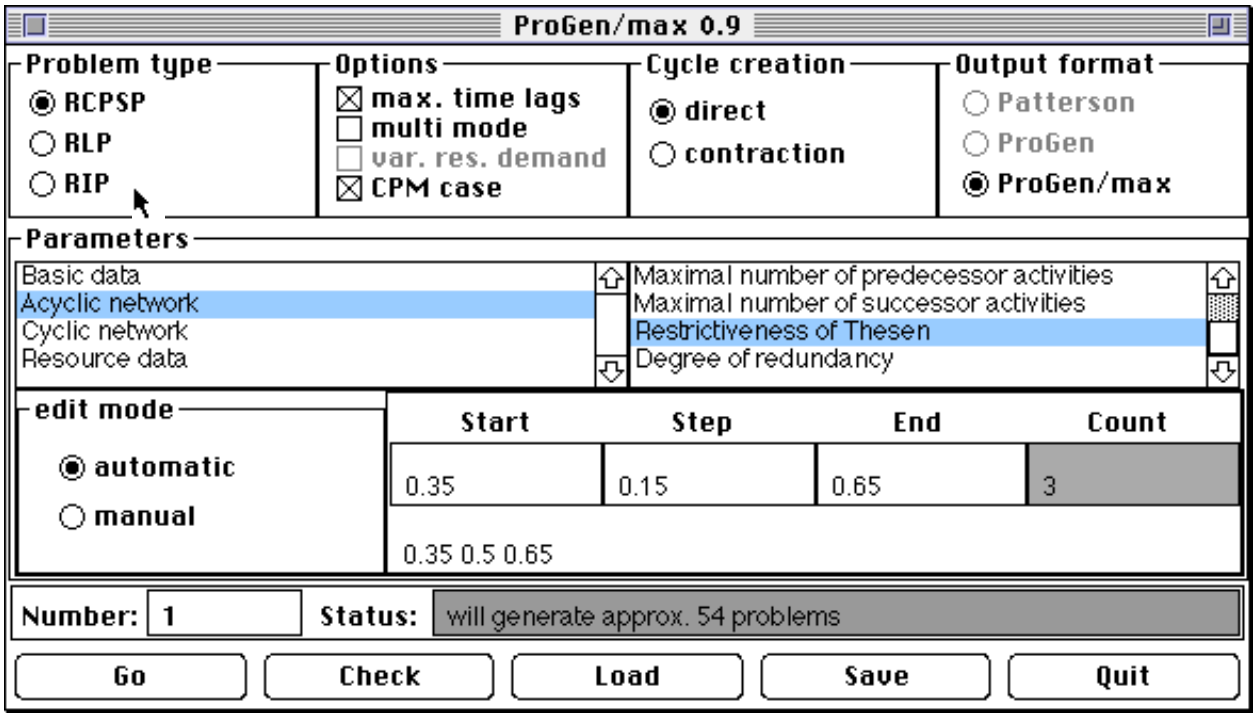


Fig. 4. ProGen / max control menue

First, the type of project scheduling problem (project duration problem RCPSP or generalization, resource leveling problem RLP or generalization, or resource investment problem RIP or generalization) has to be specified. Second, several options can be chosen: generation of maximal time lags, generation of several execution modes per activity (possibly varying w.r.t. resource demand levels), and fixing of all minimal time lags to the duration of the positive incident activity in the project network (CPM case). In case of maximal time lags, the direct or the contraction method for the creation of cycle structures can be chosen. The generated problems are filed out in the Patterson, the ProGen, or the ProGen / max format. The control parameters defined in Section 3 can be set using an automatic or a manual value editor for the selected parameter. Finally, the number of problem instances with identical parameter combination has to be specified.

The following actions can be performed:

Go: Starts the generation of all problem instances corresponding to the specified parameter configuration. The instances are labeled consecutively beginning with 1.

Check: Checks several consistency conditions for the parameter values.

Load: Sets the values in the control menue to a previously saved parameter configuration.

Save: Saves the current parameter configuration in an ASCII file (cf. Figure 5).

Quit: Exits ProGen / max.

```
7 November 1996, 10:35:17 am

Problem type: #RCPSP
Maximal time lags: true
Multi mode: false
Mode-varying resource demand: false
CPM-case: true
Cycle creation: #direct
Output format: #ProGenMax
Number: 1
Minimal number of activities: 100
Maximal number of activities: 100
Minimal duration of activity: 5
Maximal duration of activity: 15
Minimal number of initial activities: 3
Maximal number of initial activities: 7
Minimal number of terminal activities: 3
Maximal number of terminal activities: 7
Maximal number of predecessor activities: 5
Maximal number of successor activities: 5
Restrictiveness of Thesen: 0.35 0.5 0.65
Degree of redundancy: 0
Minimal percentage of maximal time lags: 0.05
Maximal percentage of maximal time lags: 0.15
Minimal number of cycle structures: 2 6
Maximal number of cycle structures: 5 9
Minimal number of nodes per cycle structure: 2
Maximal number of nodes per cycle structure: 15
Coefficient of cycle structure density: 0.3
Tightness of maximal time lags: 0.5
Slack factor: 0.0
Minimal number of resources: 5
Maximal number of resources: 5
Minimal number of resources used per activity: 1
Maximal number of resources used per activity: 5
Minimal demand of resources per activity: 1
Maximal demand of resources per activity: 3
Minimal resource factor: 0.5 0.75 1
Maximal resource factor: 0.5 0.75 1
Minimal resource strength: 0.2 0.5 0.7
Maximal resource strength: 0.2 0.5 0.7
```

Fig. 5. ProGen / max control parameters file

During the generation of problem instances, ProGen / max generates three type of files.

*.rcp, *.pro, *.pgm: problem instance in the Patterson, the ProGen, and the ProGen / max format, respectively (cf. Figure 6)

Protocol.txt: protocol file which reports on deviations of the generated instances from the given control parameters (cf. Figure 7)

Stat.txt: statistics file which contains a large number of instance characteristics like control parameters, lower bounds, and ex-post parameters

The files listed above are described in more detail in the readme.txt file which comes along with the ProGen / max image.

```
10  5     0     0
0   1     4     3     4     2     1     [0]   [0]   [0]   [0]
1   1     2     11    6     [15]  [21]
2   1     2     11    9     [15]  [24]
3   1     2     8     7     [-11] [13]
4   1     3     8     10    7     [-2]  [0]   [10]
5   1     2     1     10    [-26] [12]
6   1     1     5     [1]
7   1     1     11    [11]
8   1     1     11    [14]
9   1     2     11    2     [13]  [-24]
10  1     1     11    [14]
11  1     0
0   1     0     0     0     0     0     0
1   1     15    0     0     0     3     0
2   1     15    3     1     3     0     0
3   1     12    3     1     0     0     0
4   1     5     3     0     0     3     0
5   1     11    0     0     2     0     2
6   1     14    0     1     0     0     3
7   1     11    3     0     2     0     3
8   1     14    2     1     0     2     3
9   1     13    0     3     0     1     0
10  1     14    0     2     3     0     2
11  1     0     0     0     0     0     0
3   4     3     5     3
```

Fig. 6. RCPSP / max instance file in ProGen / max format

```
:TestSetSOR96:PSP1: OK

:TestSetSOR96:PSP2: OK

:TestSetSOR96:PSP3: OK

:TestSetSOR96:PSP4: OK

:TestSetSOR96:PSP5
Restrictiveness more than 10.0 % too large: 0.311111 > 0.25

:TestSetSOR96:PSP6: OK

:TestSetSOR96:PSP7: OK

:TestSetSOR96:PSP8: OK

:TestSetSOR96:PSP9: OK

:TestSetSOR96:PSP10: OK
```

Fig. 7. ProGen / max protocol file

References

Agrawal, M., Elmaghraby, S., Herroelen, W. (1994): DAGEN: A Generator of Testsets for Project Activity Nets; *Working Paper*, North Carolina State University, USA

Bein, W., Kamburowski, J., Stallmann, M. (1992): Optimal reduction of two-terminal directed acyclic graphs; *SIAM J. Comput. 21*, 1112 – 1129

Berge, C. (1985): *Graphs*; 2nd rev. ed., North-Holland, Amsterdam

Bondy, J.A., Murty, U.S.R. (1976): Graph Theory with Applications; The MacMillan Press, London

Davis, E.W. (1969): An Exact Algorithm for the Multiple Constrained-Resource Project Scheduling Problem; *Ph.D. Thesis*; Yale University

Davis, E.W. (1975): Project Network Summary Measures Constrained-Resource Scheduling; *AIIE Trans. 7*, 132 – 142

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

Demeulemeester, E.L. (1992): Optimal Algorithms for Various Classes of Multiple Resource-Constrained Project Scheduling Problems; *Ph.D. Thesis*, University of Leuven

Demeulemeester, E.L., Dodin, B., Herroelen, W.S. (1993): A random activity network generator; *Oper. Res. 41*, 972 – 980

de Reyck, B. (1995): On the Use of the Restrictiveness as a Measure of Complexity for Resource-Constrained Project Scheduling; *Onderzoeksrapport Nr. 9535*; Department of Applied Economics, Katholieke Universiteit Leuven

de Reyck, B., Herroelen, W.S. (1994): On the Use of the Complexity Index as a Measure of Complexity in Activity Networks; Onderzoeksrapport; Department of Applied Economics, Katholieke Universiteit Leuven

de Reyck, B., Herroelen, W.S. (1996): A Branch-and-Bound Procedure for the Resource-Constrained Project Scheduling Problem with Generalized Precedence Constraints; Onderzoeksrapport Nr. 9613; Department of Applied Economics, Katholieke Universiteit Leuven

Elmaghraby, E., Herroelen, W.S. (1980): On the measurement of complexity in activity networks; *EJOR 5*, 223 – 234

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

Franck, B., Schwindt, C. (1996): Different Resource-Constrained Project Scheduling Problems - Models and Practical Applications; *Report WIOR-450*, Institut für Wirtschaftstheorie und Operations Research, University of Karlsruhe

Kaimann, R.A. (1974): Coefficient of network complexity; *Mgmt Sci. 21*, 172 – 177

Kolisch, R., Sprecher, A., Drexl, A. (1995): Characterization and Generation of a General Class of Resource-Constrained Project Scheduling Problems; *Mgmt Sci. 41*, 1693 – 1703

Kurtulus, I., Davis, E.W. (1982): Multi-project scheduling: Categorization of heuristic rules performance; *Mgmt Sci. 28*, 161 – 172

Kurtulus, I.S., Narula, S. (1985): Multi-project scheduling: Analysis of project performance; *IIE Trans. 17*, 58 – 65

Mingozzi, A., Maniezzo, V., Ricciardelli, S., Bianco, L. (1994): An Exact Algorithm for Project Scheduling with Resource Constraints Based on a New Mathematical Formulation; *Working Paper*, University of Bologna, Italy

Neumann, K., Morlock, M. (1993): *Operations Research*; Hanser, München

Neumann, K., Schwindt, C. (1995): Projects with Minimal and Maximal Time Lags: Construction of Activity-on-Node Networks and Applications; *Report WIOR-447*, Institut für Wirtschaftstheorie und Operations Research, University of Karlsruhe

Patterson, J.H. (1976): Project scheduling: The effects of problem structure on heuristic performance; *Nav. Res. Log. Quart. 23*; 95 – 123

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

Patterson, J.H., Huber, W.D. (1974): A horizon-varying, zero-one approach to project scheduling; *Mgmt Sci. 20*, 990-998

Schrage, L. (1979): A more portable FORTRAN random number generator; *ACM Transact. Math. Software 5*, 132 – 138

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

Thesen, A. (1977): Measures of the restrictiveness of project networks; *Networks 7*, 193 – 208