Kolisch, Rainer; Sprecher, Arno; Drexl, Andreas

Characterization and generation of a general class of resource-constrained project scheduling problems: Easy and hard instances

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 301

Nr. 301

Characterization and Generation of a General Class of Resource-Constrained Project Scheduling Problems: Easy and Hard Instances

Rainer Kolisch/Arno Sprecher/Andreas Drexl

Dezember 1992

Rainer Kolisch, Arno Sprecher, Andreas Drexl, Institut für Betriebswirtschaftslehre, Christian–Albrechts–Universität zu Kiel, Olshausenstr. 40, 2300 Kiel 1, F.R.G.

Abstract: The paper describes an algorithm for the generation of a general class of precedence- and resource-constrained scheduling problems. Easy and hard instances for the single- and multi-mode resource-constrained project scheduling problem are benchmarked by using the state of the art (branch-and-bound-) procedures. The strong impact of the chosen parametric characterization of the problems is shown via an in-depth computational study. The results provided, demonstrate that the classical benchmark instances used by several researchers over decades belong to the subset of the very easy ones. In addition it is shown that hard instances, being far more smaller in size than presumed in the literature, may not be solved to optimality even within a huge amount of computational time.

Keywords: Project scheduling, precedence- and resource-constraints, nonpreemptive case, single-mode, multiple-modes, project generator, branch-and-bound methods, easy and hard instances.

1 Introduction

From the beginning of resource-constrained project scheduling research, rapid progress regarding models and methods has been documented in the literature (cf. [2], [3], [8], [12], [16], [17], [25], [41], [43], [55], [57], and [58]). But at the same time very little research concerned with the systematic generation of benchmark instances has been published. In [23] only a generator for random project scheduling problems is given. Unfortunately it does not allow to create instances subject to certain project characteristics. Hence for experimental purposes many researchers have generated their own test problems; sometimes utilizing a very restricted subset of project characteristics. Some of this work is rather well documented (cf. [12], [25], [29], [31], [44]; [48], [54]), while most efforts are only briefly described (cf. [1], [6], [7], [9], [11], [13], [14], [19], [20], [30], [32], [33], [36], [37], [42], [46], [50], [56], [59], [61], and [64]). As a result, only a few commonly used benchmark instances are available. In 1984 Patterson compared four exact procedures for makespan minimization of the single-mode resource-constrained project scheduling problems (cf. [38]). These 110 problems have been (partly) used by [4], [5], [12], [15], [17] [28], [37], [39], [40], [46], [47] and [58] and therefore became a quasi standard. Nevertheless there are three main drawbacks:

- As a collection of problems from different sources, the problems are not generated by using a controlled design of specified parameters.
- Only the single-mode case and makespan minimization is taken into consideration.

- Recent advances (cf. [17]) in the development of exact single-mode procedures have demonstrated that the Patterson-set is solvable within an average CPU-time of less than a second on a personal computer. Since there are instances (with the same number of activities) which are much more difficult to solve, they cannot be considered as a benchmark anymore.

Therefore the intention of the paper is twofold (cf. [24]): First we present an instance generator for a broad class of project scheduling problems which utilizes several parameters. Some of them have been proposed in the former literature, others are entirely new. Second we present sets of instances for the single- and the multi-mode case of the resource-constrained project scheduling problem. Solving these problems with the state of the art procedures, the strong impact of the parameters specified is demonstrated. Both the project generator PROGEN and the 1216 instances are available from the authors upon request.
The remainder of the paper is organized as follows: In section 2 we give a formal description of the model. The employed parameters and their realization within the project generator is dealt with in sections 3 and 4. The effect of the parameters used in the computational study of the single- and multi-mode case, respectively, is outlined in section 5. Some conclusions can be found in section 6. Finally a functional description of the generator is given in the appendix.

2 Notation and Model Description

We consider P projects, where each project has a specific release date ρ_p as well as a due date δ_p. The overall (super-)project consists of J partially ordered jobs, where j=1 (j=J) is the unique dummy source (sink). For the sake of simplicity project refers to the overall (super-) project as well. P_j (S_j) is the set of immediate predecessors (successors) of job j. The jobs are numerically labeled, i.e. a predecessor of j has a smaller job number than j. The precedence relations between the jobs can be represented by an acyclic activity-on-node network (AON). Furthermore the jobs within the projects are consecutively labeled with FJ_p (LJ_p) being the first (last) job of project p. Thus project p consists of $LJ_p - FJ_p + 1$ jobs.
Following the categorization scheme proposed by Slowinski (cf. [51], [52]) and Weglarz (cf. [62], [63]) we distinguish three types of (scarce) resources: the set R of renewable resources, the set N of nonrenewable resources and finally the set D of doubly constrained resources. Each resource $r \in R$ has a constant period capacity of K_r^ρ and each resource $r \in N$ has an overall capacity of K_r^ν units. Doubly constrained resources $r \in D$ are limited with respect to period capacity K_r^ρ and total capacity K_r^ν. Each job j can be processed in one of M_j modes. Job j performed in mode m has a non splittable duration of d_{jm} periods. It uses k_{jmr}^ρ units of the renewable (doubly constrained) resource r each period it is in process and consumes k_{jmr}^ν units of the nonrenewable (doubly constrained) resource r. Table 1 provides a summary of the notations and

$p = 1, \ldots, P$	:	projects
$\rho_p(\delta_p)$	:	release date (due date) of project p
c_p	:	cost incurring per period project p is finished after its due date
$FJ_p(LJ_p)$	:	number of the first (last) job of project p
$j = 1(J)$	:	unique source (sink) of the network
$P_j(S_j)$	:	set of immediate predecessors (successors) of job j
$EF_j(LF_j)$	:	earliest (latest) finish time of job j
$\bar{T}$	:	upper bound on the projects makespan (horizon)
$r \in R\,(N, D)$	:	set of renewable (nonrenewable, doubly constrained) resources
$m = 1, ..., M_j$	:	modes of job j
d_{jm}	:	(non preemptable) duration of job j scheduled in mode m
k^{ρ}_{jmr}	:	per period usage of renewable (doubly constrained) resource r required to perform job j in mode m
k^{ν}_{jmr}	:	total consumption of nonrenewable (doubly constrained) resource r required to perform job j in mode m
K^{ρ}_r	:	per period availability of renewable (doubly constrained) resource r
K^{ν}_r	:	total availability of nonrenewable (doubly constrained) resource r

Table 1: Symbols and Definitions

definitions.

For modelling purposes we use binary variables as proposed in [43] for $j = 1, \ldots, J$, $m = 1, \ldots, M_j$, $t = EF_j, \ldots, LF_j$:

$$x_{jmt} = \begin{cases} 1 & , \text{ if job j is performed in mode m and completed in period t} \\ 0 & , \text{ otherwise.} \end{cases}$$

The constraints are given in Table 2. (1) ensures that each job is assigned exactly one mode and a completion time within its time window $[EF_j, LF_j]$. The time window of feasible finish times is calculated by forward and backward recursion as shown in [19]. (2) indicates that no job starts before the release date of its project while (3) warrants that no job ends after the due date of its project. Precedence relations between related jobs are maintained by (4). (5) secures feasibility with respect to renewable and doubly constrained resources. Finally (6) limits the consumption of the nonrenewable and doubly constrained resources to their

$\sum_{m=1}^{M_j} \sum_{t=EF_j}^{LF_j} x_{jmt} = 1$	$j = 1, \ldots, J$	(1)
$\sum_{m=1}^{M_j} \sum_{t=EF_j}^{LF_j} (t - d_{jm})\, x_{jmt} \geq \rho_p$	$p = 1, \ldots, P,\ j = FJ_p, \ldots, LJ_p$	(2)
$\sum_{m=1}^{M_j} \sum_{t=EF_j}^{LF_j} t\, x_{jmt} \leq \delta_p$	$p = 1, \ldots, P,\ j = FJ_p, \ldots, LJ_p$	(3)
$\sum_{m=1}^{M_h} \sum_{t=EF_h}^{LF_h} t\, x_{hmt} \leq \sum_{m=1}^{M_j} \sum_{t=EF_j}^{LF_j} (t - d_{jm})\, x_{jmt}$	$j = 1, \ldots, J,\ h \in P_j$	(4)
$\sum_{j=1}^{J} \sum_{m=1}^{M_j} k_{jmr}^{\rho} \sum_{q=t}^{t+d_{jm}-1} x_{jmq} \leq \kappa_{rt}^{\rho}$	$r \in R \cup D,\ t = 1, \ldots, \bar{T}$	(5)
$\sum_{j=1}^{J} \sum_{m=1}^{M_j} k_{jmr}^{\nu} \sum_{t=EF_j}^{LF_j} x_{jmt} \leq \kappa_r^{\nu}$	$r \in N \cup D$	(6)
$x_{jmt} \in \{0, 1\}$	$j = 1, \ldots, J,\ m = 1, \ldots, M_j,$ $t = EF_j, \ldots, LF_j$	(7)

Table 2: Constraints

availability.

The most common objective function w.r.t. (1)-(7) is the makespan minimization

$$\text{minimize} \quad \sum_{m=1}^{M_J} \sum_{t=EF_J}^{LF_J} t\, x_{Jmt}.$$

Another objective is, e.g. the minimization of the weighted project delay

$$\text{minimize} \quad \sum_{p=1}^{P} c_p \left(\max_{j=FJ_p}^{LJ_p} \left\{ \sum_{m=1}^{M_j} \sum_{t=EF_j}^{LF_j} t\, x_{jmt} \right\} - \delta_p \right)^+,$$

where

$$z^+ := \begin{cases} z & , \text{ if } z \geq 0 \\ 0 & , \text{ otherwise.} \end{cases}$$

This formulation embodies a wide range of precedence- and resource-constrained scheduling problems, especially the single- ($P = 1$, $M_j = 1$, $j = 1, \ldots, J$, $N = D = \emptyset$, $\rho_p = 0$, $\delta_p = \bar{T}$) and the multi-mode problem ($P = 1$, $\rho_p = 0$, $\delta_p = \bar{T}$) of resource-constrained project scheduling. Furthermore job shop and flow shop type problems as well as scheduling problems with one and multiple parallel machines are included. Note

that the main emphasis of the paper is on the generation of the set of solutions, i.e. the constraints (1)-(7). In addition it is easy to incorporate other (regular) objective functions. Details are left to the reader (and user of PROGEN).

3 Project Generation

3.1 Basedata Generation

In this section we briefly outline the generation of the projects basedata. We use the functions round and trunc as well as the random functions rand and $\overline{\text{rand}}$ defined as follows:

$\text{rand}[n_1, n_2]$: integer random number out of the interval $[n_1, n_2]$

$\overline{\text{rand}}[n_1, n_2]$: real random number out of the interval $[n_1, n_2]$.

The (pseudo) random numbers are constructed by transforming $[0, 1)$ uniformly distributed random numbers. The $[0, 1)$ uniformly distributed random numbers are calculated via the congruence-generator developed by Lehmer using the constants and implementation as given in [49]. The generation of the basedata needs no further explanation. The input and output is displayed in Tables 3 and 4, respectively. MPM_p denotes the MPM-duration of project p, $p = 1, \ldots, P$. It is calculated with respect to the release dates by using the modes of shortest duration and the network, the construction of which is described in the next section.

P	:	number of projects
$J^{min}(J^{max})$	:	minimal (maximal) number of jobs per project
$M^{min}(M^{max})$	:	minimal (maximal) number of modes per job
$d^{min}(d^{max})$	:	minimal (maximal) duration per job
ρ^{max}	:	maximal release date
δ_{fac}	:	due date factor $\in [0, 1]$

Table 3: Input Basedata Generation

3.2 Network Generation

In section 2 we stated that the structure of the project can be depicted as an acyclic AON. Thus it is a quite natural approach to construct the network by using the following simple implication of the definition of a network:

J_p	:=	$\text{rand}[J^{min}, J^{max}], p = 1, \ldots, P$
	=	number of jobs of project p
J	:=	$\sum_{p=1}^{P} J_p + 2$
	=	total number of jobs (including super-source and -sink). The jobs are numerically and consecutively labeled within the projects. That is, project p consists of the numerically labeled jobs j, $j = \sum_{q=1}^{p-1} J_q + 2, \ldots, \sum_{q=1}^{p} J_q + 1$.
M_j	:=	$\text{rand}[M^{min}, M^{max}], j = 2, \ldots, J-1, (M_1 = M_J = 1)$
	=	number of modes of job j
d_{jm}	:=	$\text{rand}[d^{min}, d^{max}], j = 2, \ldots, J-1, m = 1, \ldots, M_j$ $(d_{11} = d_{J1} = 0)$. The modes are labeled with respect to non-decreasing durations.
ρ_p	:=	$\text{rand}[0, \rho^{max}]$
	=	release date of project p
$\bar{T}$	:=	$\max_{p=1}^{P} \rho_p + \sum_{j=1}^{J} \max_{m=1}^{M_j} \{d_{jm}\}$
	=	horizon
δ_p	:=	$\text{trunc}(\text{MPM}_p + \delta_{fac}(\bar{T} - \text{MPM}_p))$
	=	due date of project p
c_p	:=	$\text{trunc}(\overline{\text{rand}}[0,1] * J_p)$
	=	per period tardiness costs of project p

Table 4: Output Basedata Generation

Theorem 1 *(cf. [35], p.33)*
Let $N = (V, A)$ be a network with node set V and arc set A. Then, for every node $v \in V$ there is a directed path from the single source to v and a directed path from v to the single sink.

That is, every node except of the sink (source) has at least one successor (predecessor). Therefore the basic idea is as follows: First, determine one predecessor for each node, second, determine one successor for each node and then add further arcs.

We consider the example in Figure 1 (cf. [21], p. 179), where the additional arc $(2, 7)$ would give no extra information about scheduling the activities and therefore should not be taken into consideration. We use

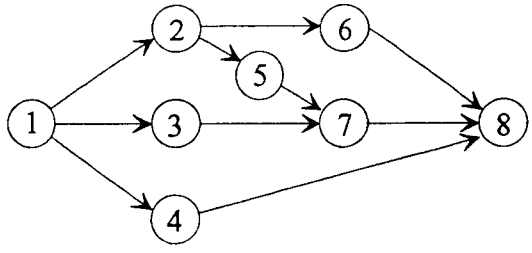

Figure 1: Example Network

the following definition:

Definition 1

Let $N = (V, A)$ be a network. An arc (h, j) is called redundant, if there are arcs $(i_0, i_1), \ldots, (i_{s-1}, i_s) \in A$ with $i_0 = h$, $i_s = j$ and $s \geq 2$.

That is, an arc (i, j) is redundant, if it is an element of the transitive closure $\bar{N}^+$ of $\bar{N} = (V, A \backslash \{(i, j)\})$. If within the construction process of the network an arc (i, j) is chosen for adding it to the actual graph, four cases of redundancy might occur (cf. Figure 2, where $\bar{N} = (V, A)$ denotes the current graph with actual sets of (immediate) successors $\bar{S}_j(S_j)$ and (immediate) predecessors $\bar{P}_j(P_j)$). For a given cardinality of the set

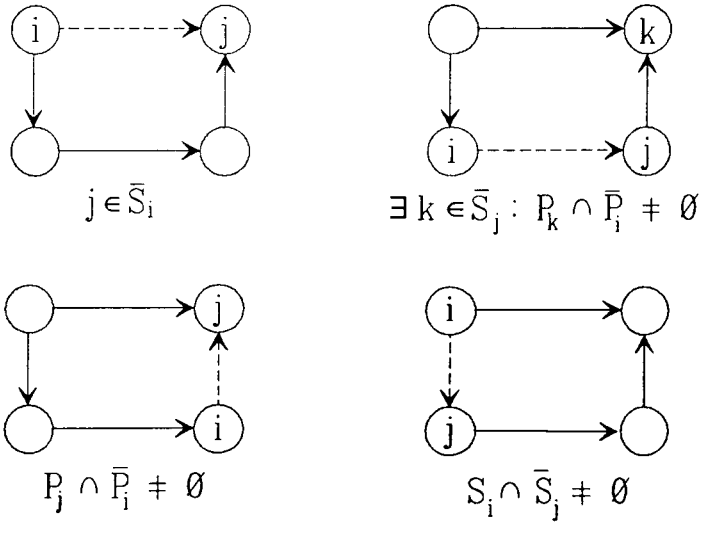

Figure 2: Cases of Redundancy

of nodes the mimimal and maximal number of non-redundant arcs are given in the following theorem and illustrated in Figures 3 and 4.

Theorem 2

Let $N = (V, A)$ be a network with $|V| = n$.

(a) Since a network is connected, the minimal number of non-redundant arcs A^{min} is given by

$$A^{min} = n - 1.$$

(b) The maximal number of non-redundant arcs A^{max} in a network with $n \geq 6$ is given by

$$A^{max} = \begin{cases} n - 2 + \left(\frac{n-2}{2}\right)^2 & : \text{ if } n \text{ is even} \\ n - 2 + \left(\frac{n-1}{2}\right)\left(\frac{n-3}{2}\right) & : \text{ if } n \text{ is odd.} \end{cases}$$

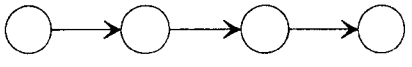

Figure 3: Minimal Number of Non-redundant Arcs

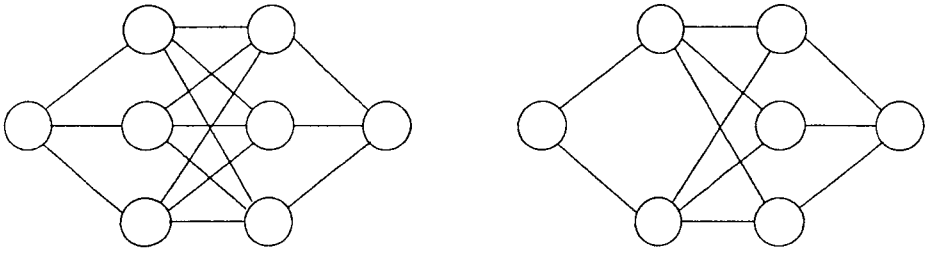

Figure 4: Maximal Number of Non-redundant Arcs

For the characterization of the network we use the parameters given in Table 5. The complexity as the average number of (non-redundant) arcs per node is a measure for the network logic, which has been introduced by Pascoe (cf. [36]) for activity-on-arc networks and adopted by Davis (cf. [13]) for the AON representation. For the latter complexity has to be understood in the way that for a fixed number of jobs a higher complexity results in an increasing number of arcs and therefore in a greater interconnectedness of the network. It has already been shown by Alvarez-Valdes/Tamarit (cf. [1]) and will be confirmed in this study that with increasing complexity problems become easier. This makes the term complexity somewhat confounding. Nevertheless we stay with the term, because it has been used in a lot of computational studies (cf. [54], [37], [57], [31] and [20]) and has become a well known project summary measure. Two disadvantages associated

S_1^{min} (S_1^{max})	:	minmimal (maximal) number of start activities
P_J^{min} (P_J^{max})	:	minmimal (maximal) number of finish activities
S_j^{max} (P_j^{max})	:	maximal number of successor (predecessor) activities of activity j, $j = 2, \ldots, J-2$
C	:	network complexity, i.e. the average number of non-redundant arcs per node (including the super-source and -sink)
ϵ_{NET}	:	tolerated complexity deviation

Table 5: Input Network Generation

with this measure have to be mentionend - to wit:

(i) The number of arcs only does not give all informations about the number of possible schedules. Attempts in order to find more elaborate measures than complexity can be found in [26], [60] and [22]. But as pointed out by Elmaghraby and Herroelen (cf. [22]) "it seems evident to us that the structure of the network - in whichever way it is measured - will not be sufficient to reflect the difficulty encountered in the resolution of such problems".

(ii) The measure is not normalized to the interval [0,1]. A normalized measure for the network structure is the "Order Strength" which has been proposed by Mastor (cf. [34]) for the assembly line balancing problem and used by Cooper (cf. [10]) for the project scheduling problem. The Order Strength for AON-representation is calculated by dividing the number of arcs by the maximal number of arcs which is n(n-1)/2. Unfortunately the maximal number of arcs has two drawbacks: It includes redundant arcs and is far greater then a realistic number of precedence relations within scheduling problems. Although we can use the maximal number of non-redundant arcs for normalizational purpose, they still exceed the number of realistic precedence relations. As a consequence for realistic projects the order strength converges to zero with an increasing number of jobs.

We now describe the network construction for a single project (Figure 5), a multi-project network is maintained analogously. In Step 1 the number of start- and finish-activities are drawn randomly out of the interval $[S_1^{min}, S_1^{max}]$ and $[P_J^{min}, P_J^{max}]$, respectively. Then, the arcs, which connect the dummy source with the start activities and the finish-activities with the dummy sink are added to the network. In Step 2, beginning with the lowest indexed non-start activity, each activity is assigned a predecessor (activity) at random. Similar in Step 3, where each activity, which has no successor, is assigned one, cf. arcs (3,6) and (6,9) in Figure 5. In both steps the jobs are considered in order of increasing job number. Finally (in Step 4) further arcs are

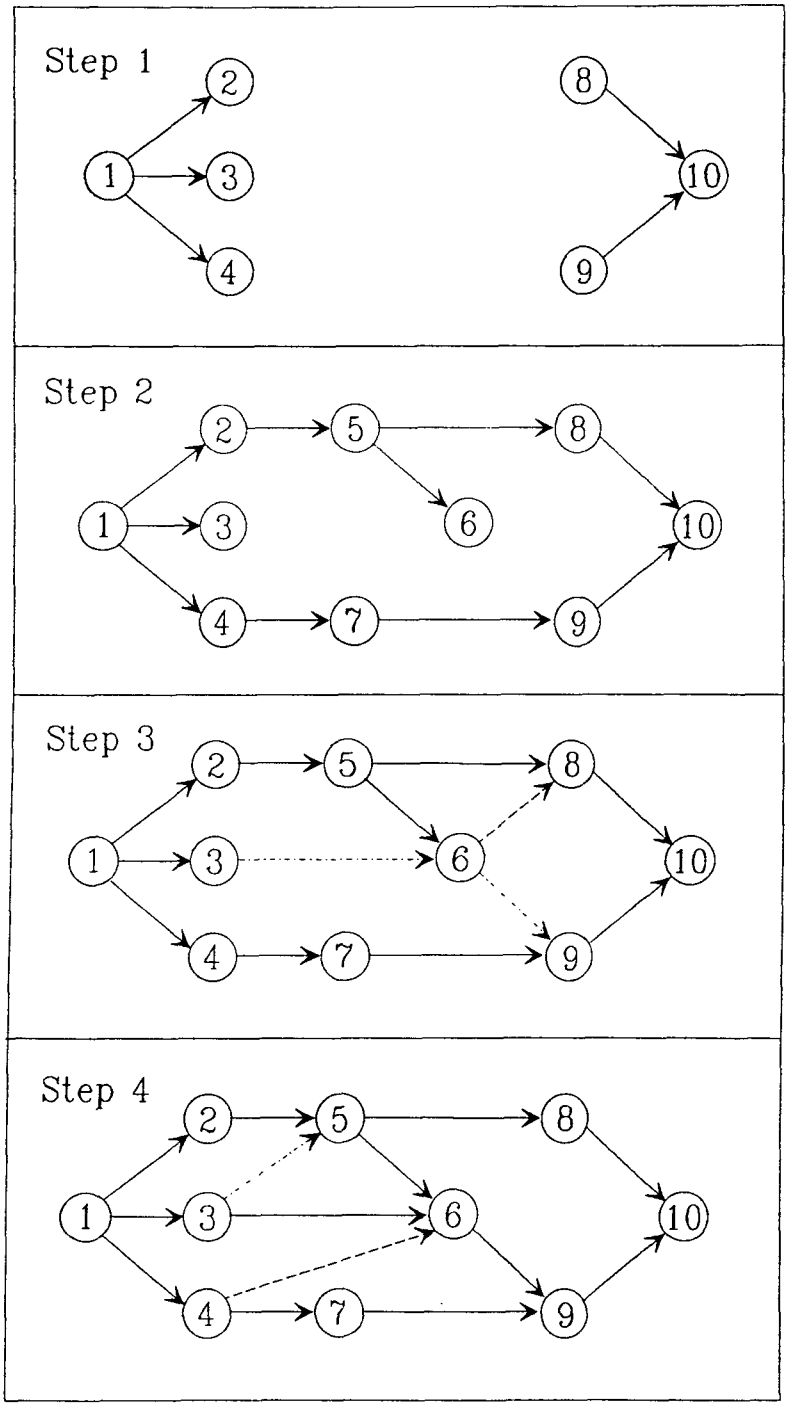

Figure 5: Network Generation

added until the complexity is reached. During the whole procedure one has to take into account:

- To avoid redundancy, there must be no precedence relations between the start-activities and the finish-activities, respectively.
- Adding arcs in Step 3 (e.g. arc (6,8)) or 4 must not produce redundant precedence relations.
- The limitation given by the maximal number of successors and predecessors and the number of start and finish activities (e.g. arc (4,6) in Step 4, which cannot be incorporated, if at most two predecessors are allowed).

In the following cases the generation procedure has to be restarted:

- If the required complexity is low, i.e. $C \approx 1$, it might happen that after Step 3 the number of arcs integrated into the network is too high, that is,

  $$\text{ActArcs} \quad > \quad J * C * (1 + \epsilon_{NET}).$$

- If in Step 3, due to the limited number of predecessors, there is no successor of a job j available.
- If in Step 3 for a job j, there are only successors available, which lead to redundant precedence relations.
- If the required complexity is not obtainable in Step 4, that is, within a limited number of trials of randomly selecting a node and calculating possible successors, there are no further arcs addable to obtain

  $$ActArcs \quad \geq \quad J * C * (1 - \epsilon_{NET}).$$

By an appropriate reduction of the set of choosable predecessors and successors in the steps previously described a numerical labeled network is realized.
Through adjustment of the input parameters special network structures, e.g. general (Figure 6), serial structures (Figure 7) and network shapes as described in [30], [31] and [53] are obtainable.

4 Resource Demand and Availability Generation

4.1 Resource Demand Generation

The resource demand generation consists of two decisions to be made. First, we have to determine the resources used or consumed by the job-mode combinations [j,m], $j = 1, \ldots, J$, $m = 1, \ldots, M_j$. Second, if a job-mode combination uses or consumes a resource, we have to calculate the number of units used or

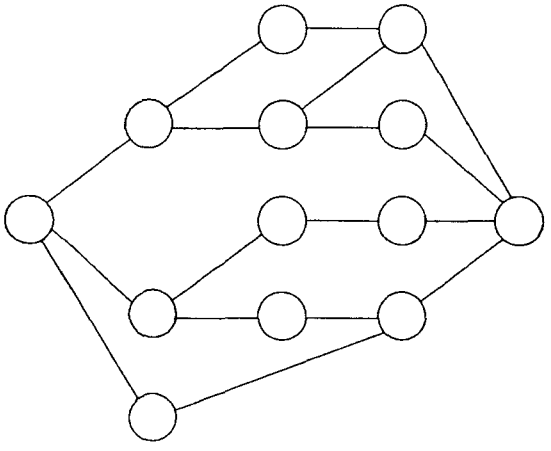

Figure 6: Multi-Project with General Structure

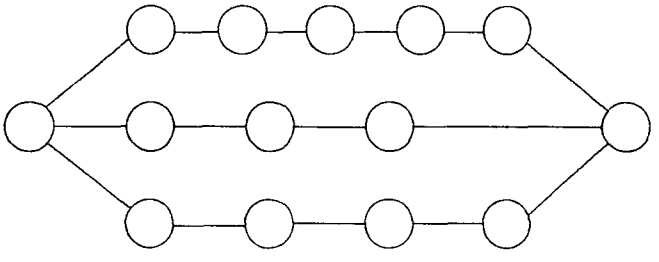

Figure 7: Multi-Project with Serial Structure

consumed. To the first step we refer with *request generation* (Subsection 4.1.1) and to the latter we refer with *generation of demand level* (Subsection 4.1.2).

We consider a resource type $\tau \in \{R, N, D\}$. The number of resources of type τ is determined by a randomly drawn integer within $[|\tau|^{min}, |\tau|^{max}]$, that is

$$|\tau| \quad := \quad \text{rand}[|\tau|^{min}, |\tau|^{max}].$$

4.1.1 Requested Resources

For characterizational purposes we use a generalization of the resource factor (RF) which has been introduced by Pascoe (cf. [36]) for the single-mode case and which has later on been utilized in studies by Cooper (cf. [10]) and Alvarez-Valdes/Tamarit (cf. [1]). For the single-mode case RF is calculated as follows:

$$RF \quad := \quad \frac{1}{J}\frac{1}{|R|}\sum_{j=1}^{J}\sum_{r\in R}\begin{cases} 1 & , \text{ if } k_{jr} > 0 \\ 0 & , \text{ otherwise.} \end{cases}$$

$\|\tau\|^{min}(\|\tau\|^{max})$	:	minimal (maximal) number of resources of type τ
$Q_\tau^{min}(Q_\tau^{max})$	:	minimal (maximal) number of resources of type τ used by a job-mode combination $[j,m]$
$U_\tau^{min}(U_\tau^{max})$	:	minimal (maximal) demand for a resource of type τ
$P_\tau(F=1)(P_\tau(F=2))$	:	probability that demand for a resource of type τ is duration constant (monotonically decreasing with the duration)
RF_τ	:	resource factor of type τ
RS_τ	:	resource strength of type τ
ϵ_{RF}	:	tolerated resource factor deviation

Table 6: Input Demand Generation

The resource factor reflects the average portion of resources requested per job. It is a measure of the density of the array $k_{j\tau}$. If we have RF=1, then each job requests all resources. RF=0 indicates that no job requests any resource, thus we obtain the unconstrained MPM-case. In order to use RF for the multi-mode case as well, we generalize it straightforward to a type dependent resource factor RF_τ, $\tau \in \{R, N, D\}$:

$$RF_\tau := \frac{1}{J-2}\frac{1}{|\tau|}\sum_{j=2}^{J-1}\frac{1}{M_j}\sum_{m=1}^{M_j}\sum_{r\in\tau}\begin{cases} 1 & , \text{ if } k_{jmr} > 0 \\ 0 & , \text{ otherwise.} \end{cases}$$

Again RF is normalized to the interval [0,1] with the interpretation very close to the one of the original RF. It reflects the average portion of resources out of one type, requested by each job-mode combination [j,m] and it measures the density of the three dimensional array k_{jmr}. Of course, our RF equals the one proposed by Pascoe for the case $N = D = 0$ and $M_j = 1$, $j = 1, \ldots, J$. Table 6 shows the other input parameters as well.

For the generation of the resource request we use the following internal variables and data structures: First, we represent the information whether a job-mode combination [j,m] requests resource r by a three-dimensional array $Rq[j,m,r]$ of binary digits. Rq[j,m,r] is initialized with zeros and is set equal to one, if and only if [j,m] requests resource r. The actual resource factor (ARF) is then calculated as follows:

$$\mathrm{ARF}_\tau := \frac{1}{J-2}\frac{1}{|\tau|}\sum_{j=2}^{J-2}\frac{1}{M_j}\sum_{m=1}^{M_j}\sum_{r\in\tau} Rq[j,m,r].$$

The actual number of resources requested by [j,m] is obtained by

$$\mathrm{Q[j,m]} := \sum_{r\in\tau} Rq[j,m,r].$$

Finally we get CT, the actual set of choosable triplets,

$$CT \quad := \quad \{[j,m,r]; Rq[j,m,r] = 0 \text{ and } Q[j,m] < Q_\tau^{max}\},$$

that is, the set of job-mode-resource combinations [j,m,r], which are furthermore choosable ($Rq[j,m,r] = 0$) without $Q[j,m]$ exceeding Q_τ^{max}.

	Establishing the minimal number of resources requested by [j,m]	Establishing the resource factor
2	1 ... M_2	1 ... M_2
	1 2 3 ... 1 2 3	1 2 3 ... 1 2 3
	0 0 1 ... 1 0 0	1 0 1 ... 1 0 1
	⋮ ⋮	⋮ ⋮
$J-1$	1 ... M_{J-1}	1 ... M_{J-1}
	1 2 3 ... 1 2 3	1 2 3 ... 1 2 3
	1 0 0 ... 0 0 1	1 0 0 ... 1 0 0
	Step 1	Step 2
	$Q_\tau^{min} = 1, Q_\tau^{max} = 2$	

Table 7: Resource Factor Establishing

During the two steps to be performed the internal variables are continuously updated.
In Step 1 for each job-mode combination [j,m], as far as the minimal number of requested resources Q_τ^{min} is not reached, additional resources are selected randomly. While, in Step 2, the actual resource factor is less than the asserted one and in addition there are choosable triplets in CT, i.e. $CT \neq \emptyset$, the actual resource factor is incremented by randomly drawing a triplet out of CT. In Table 7, where we have $|\tau| = 3$, the triplet (2,1,2) is not in the choosable set CT, because Q_τ^{max} is fixed to two.
If after Step 2 the actual resource factor declines more then tolerated, i.e.

$$\text{ARF}_\tau \quad \notin \quad [RF_\tau \cdot (1 - \epsilon_{RF}), RF_\tau \cdot (1 + \epsilon_{RF})],$$

then a warning message is given.

4.1.2 Level of Demand

If we have $Rq[j,m,r] = 1$, then a positive demand of the job-mode combination [j,m] for resource r has to be generated. The interrelation between the durations of the modes and the demand for resource r is reflected

by two types of functions. One of which is duration independent ($F = 1$) and the other one is decreasing with

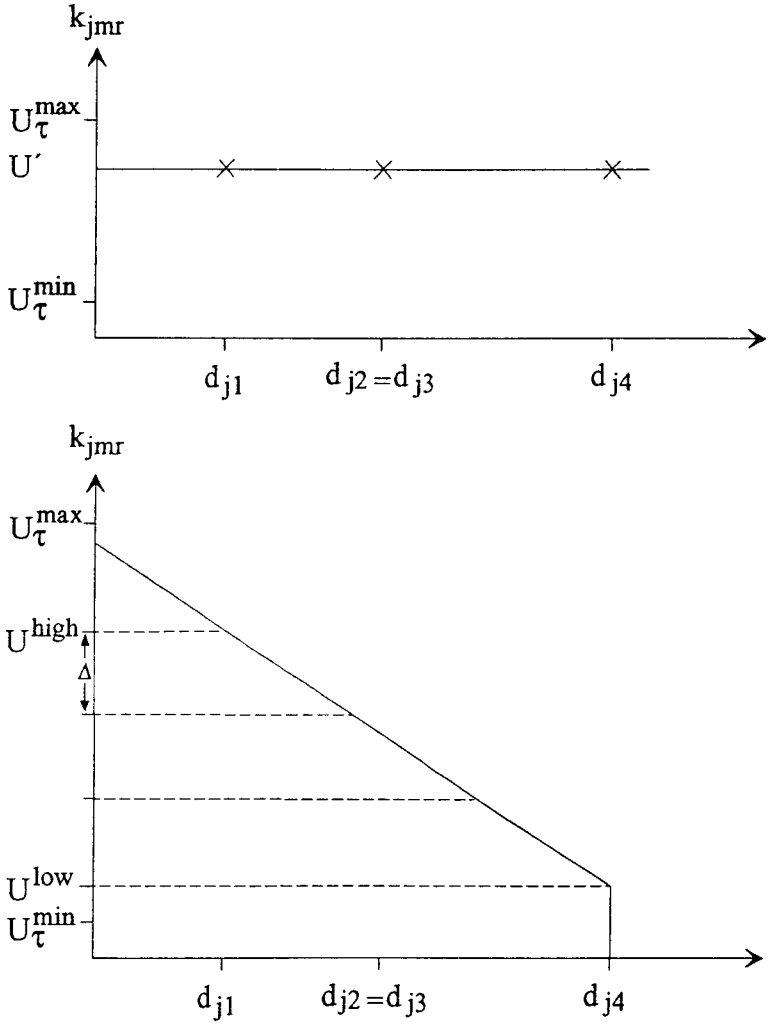

Figure 8: Interrelation between Demand and Duration

the (increasing) duration (F=2). That is, for the renewable and doubly constrained resources the per-period demand and for the nonrenewable resources the total demand is generated as the interrelation prescribes. For each resource $r \in \tau$ the interrelation is defined by

$$F_\tau(r) \quad := \quad \begin{cases} 1 & : \text{ if } \overline{\text{rand}}[0,1] < P_\tau(F=1) \\ 2 & : \text{ otherwise} \end{cases}$$

given the type dependent probabilities $P_\tau(F = 1)$ and $P_\tau(F = 2)$. If $F_\tau(r) = 1$, then for each job the demand U' is randomly drawn out of the integer interval $[U_\tau^{min}, U_\tau^{max}]$ and is then assigned to all modes,

which request this resource. In the case of $F_\tau(r) = 2$, for each job j two levels are drawn randomly out of the parameter specified interval:

$$U^1 := \text{rand}[U_\tau^{min}, U_\tau^{max}] \quad , \quad U^2 := \text{rand}[U_\tau^{min}, U_\tau^{max}].$$

Then U^{low} and U^{high} are calculated as follows

$$U^{low} := \min\{U^1, U^2\} \quad , \quad U^{high} := \max\{U^1, U^2\}$$

Let $\bar{M}_j$ be the number of modes of job j with different durations requesting resource r. We calculate

$$\Delta \quad := \quad \frac{U^{high} - U^{low}}{\bar{M}_j}$$

and yield $\bar{M}_j$ intervals I_k as follows:

$$I_k := [Round(U^{high} - \Delta k), Round(U^{high} - \Delta(k-1))] \qquad k = 1, \ldots, \bar{M}_j.$$

Since the modes are labeled with respect to nondecreasing durations, we can now draw the demand randomly out of the intervals corresponding to the durations. Figure 8 illustrates the generation of the level of demand.

Remark 1

If for $m, \bar{m} \in \{1, \ldots, M_j\}$, $m \neq \bar{m}$, it is $d_{jm} = d_{j\bar{m}}$ and $Rq[j, m, r] = 1 = Rq[j, \bar{m}, r]$, then the demand is generated randomly out of the same interval.

Due to the construction inefficiency, which is defined in the following, might occur:

Definition 2

A job j has inefficient modes, if there are modes m and $\bar{m}$ with $d_{jm} \leq d_{j\bar{m}}$ and $k^\rho_{jmr} \leq k^\rho_{j\bar{m}r}$ for all $r \in R \cup D$ and $k^\nu_{jmr} \leq k^\nu_{j\bar{m}r}$ for all $r \in N \cup D$.

If inefficient modes occur for job j, we calculate the number of resources requested by job j

$$Q_j \quad := \quad \sum_{m=1}^{M_j} \sum_{r \in \tau} Rq[j, m, r]$$

and the request and demand generation is restarted with the additional constraint

$$Q_j \quad = \quad \sum_{m=1}^{M_j} Q[j, m].$$

If efficiency is not obtainable within MaxTrials, the generation is interrupted and the parameters have to be adjusted.

4.2 Resource Availability Generation

In order to express the relationship between the resource demand of the jobs and the resource availability Cooper (cf. [10]) introduced the resource strength (RS), which is calculated as follows:

$$RS_r := \frac{K_r}{\frac{1}{J}\sum_{j=1}^{J} k_{jr}}.$$

Later the RS has been utilized by Alvarez-Valdes/Tamarit (cf. [1]). There are three main drawbacks of the proposed measure. We will point them out and propose a new RS to overcome these disadvantages:

- First, the RS is not standardized in the intverval [0,1].
- Second, a rather small RS does not guarantee a feasible solution. E.g. for three jobs with $k_{jr} = 1$, 1 and 10, respectively, one has to adjust the resource strength to $RS_r \geq 2.5$ in order to achieve a feasible solution.
- Third and most important, regard the myopic fashion in which the scarcity of resources is calculated. This shall be depicted with the following simple example: We consider two projects, with exactly the same data except the network. Project 1 has a parallel structure, where each job is immediate successor of the dummy source and immediate predecessors of the dummy sink, whereas project 2 has a serial structure, where each job has exactly one predecessor and one successor. Let us further assume that the resource availability is large enough in order to assure feasibilty of both problems. Then the RS for both projects will be exactly the same, but obviously the serially structured project, being the MPM-case, will be quite easy to solve, whereas the parallel structured project is, dependent on the amount of resource availability, rather difficult.

In order to overcome these disadvantages, we have created the following methodology for a measure of resource scarceness which is applicable to all types of resources. We determine a minimal demand K_r^{min} as well as a maximal demand K_r^{max} and let the resource availability be a convex combination of the two with RS_r as scaling parameter : $K_r := K_r^{min} + RS_r(K_r^{max} - K_r^{min})$. Thus with respect to one resource we will get the smallest feasible resource availabilty for $RS_r = 0$. For $RS_r = 1$ the amount of resources is just large enough to achieve the MPM-case.

For the nonrenewable resources r, $r \in N \cup D$, the minimal and maximal availabilities to complete the project can be calculated as follows:

$$K_r^{min} := \sum_{j=2}^{J-1} \min_{m=1}^{M_j} \{k_{jmr}^{\nu}\} \quad , \quad K_r^{max} := \sum_{j=2}^{J-1} \max_{m=1}^{M_j} \{k_{jmr}^{\nu}\}.$$

For a given type dependent resource strength $RS_\tau \in [0,1]$ the availability is

$$K_r^\nu := K_r^{min} + \text{Round}(\text{RS}_\tau\,(K_r^{max} - K_r^{min})).$$

If the considered resource is renewable the minimal demand is

$$K_r^{min} := \max_{j=2}^{J-1} \left\{ \min_{m=1}^{M_j} \{k_{jmr}^\rho\} \right\}.$$

The maximal demand is calculated as the peak demand of the precedence preserving earliest start schedule. Thereby each job is performed in the lowest indexed mode employing maximal per-period demand with respect to the resource under consideration. That is, we determine the maximal per-period demand of job j with respect to resource r

$$k_{jr}^* := \max_{m=1}^{M_j} \{k_{jmr}^\rho\}$$

and the corressponding mode with shortest duration:

$$m_{jr}^* := \min_{m=1}^{M_j} \{m | k_{jmr}^\rho = k_{jr}^*\}$$

Given the precedence relations and due dates of the project we can now calculate the earliest start schedule with the modes determined. We obtain the resource dependent start time ST_j^r and completion time CT_j^r of job j, $j = 2, \ldots, J-1$. We then calculate the peak period demand

$$K_r^{max} := \max_{t=1}^{\bar{T}} \left\{ \sum_{\substack{j=2 \\ ST_j^r + 1 \le t \le CT_j^r}}^{J-1} k_{jm_{jr}^* r} \right\}$$

and the available amount using the type dependent resource strength RS_τ

$$K_r^\rho := K_r^{min} + \text{Round}(RS_\tau (K_r^{max} - K_r^{min})). \tag{8}$$

By constuction we can state the following:

Remark 2

(a) If $|\tau| = 1$ and $RS_\tau = 0$, then the lowest resource feasible level with respect to τ will be generated.

(b) For $RS_\tau = 1$ the resource unconstrained MPM-case with respect to τ will be generated.

(c) IF $RS_\tau << 1$ and $M_j > 1$ feasibility of the problem can not be assured, because of mode coupling via resource constraints.

5 Computational Results

5.1 Single-Mode Case

Currently the most advanced exact procedure for solving makespan minimization problems seems to be the implicit enumeration procedure of the B&B type with backtracking from Demeulemeester (cf. [17], [18]). It is coded in C and solves the fourty-three 27-job problems out of the 110 Patterson instances in an average computational time of 1.06 seconds to optimality on an IBM PS/2 Model 55sx (80386sx processor, 15 Mhz clockpulse). We used the original implementation of the algorithm provided by Demeulemeester in our computational study.

We have carried out two series of experiments for single-mode problems. First we used a full factorial design, where we varied the complexity C, the resource factor RF and the resource strength RS. The constant and the varying parameter levels are documented in Table 8 and 9, respectively. Obviously we have $|N| = |D| = 0$ and $P_R(F = 1) = 1$. Using 10 projects for each combination of C, RF and RS a total of $3 \cdot 4 \cdot 4 \cdot 10 = 480$ instances were generated. All of them were solved with the exact solution procedure. Utilizing the previously described machine we imposed a time limit of 3600 seconds on the maximal CPU time.

	J	M_j	d_j	$\|R\|$	U_R	Q_R	S_1	S_j	P_J	P_j
min	30	1	1	4	1	1	3	1	3	1
max	30	1	10	4	10	4	3	3	3	3

Table 8: Constant Parameter Levels for Single-Mode Instances under Full Factorial Design

C	1.5	1.8	2.1	
RF_R	0.25	0.5	0.75	1.0
RS_R	0.2	0.5	0.7	1.0

Table 9: Variable Parameter Levels for Single-Mode Instances under Full Factorial Design

Our 480 instances have been solved in 461.25 seconds on the average. The minimum solution time turned out to be 0.0 seconds (which is actually less than 0.05 seconds), while the maximum solution time was the imposed limit of 3600 seconds. Table 10 provides the frequency distribution of the solution times. Among the 65 very hard problems which needed more than 1000 CPU-seconds were 52 for which an optimal solution could not either be found or verified within the imposed time limit.

Range	[0,0.1]	(0.1 , 1]	(1,10]	(10,100]	(100,1000]	≥1000
Instances	165	142	46	36	26	65

Table 10: Frequency Distribution of Solution Times for Single-Mode Instances under Full Factorial Design

In order to find out the effects of the different parameters we performed a mean value analysis regarding CPU-times for each of the varying parameters.

The effects of altering the complexity C can be seen in Table 11. As C is enlarged from 1.5 to 2.1 the solution times decrease. This is due to the fact that adding more precedence relations to the network lowers the number of feasible schedules for a given upper bound on the projects makespan. This reduces the enumeration tree and makes the problems more easy. The effect has already been mentioned by Alvarez-Valdes/Tamarit for heuristics (cf. [1]).

C	1.5	1.8	2.1
μ_{CPU}	674.76	477.80	231.19

Table 11: Effects of Complexity C on Solution Times

The increase of the resource factor results in an increase of solution times (cf. Table 12). This contradicts the results of Alvarez-Valdes/Tamarit. They observed that problems with a resource factor of 1.0 were easier than ones with a resource factor of 0.5. We assume that their results were somewhat distorted through the use of a myopic resource strength, which has already been pointed out in section 4. It can be concluded that problems become harder, when the average portion of resources requested per job increases. It has to be remarked that the majority of the 110 instances of Patterson have a resource factor of 1.0.

RF_R	0.25	0.5	0.75	1.0
μ_{CPU}	0.30	128.35	787.98	928.30

Table 12: Effects of the Resource Factor RF_R on Solution Times

From Table 13 it can be seen that the resource strength has the strongest impact on solution times. Problems with a RS_R of 0.2 turned out to be the hardest. Out of those 120 instances for 47 the optimum solution could not be found or verified within the imposed time limit. The problems with a RS_R of 1.0 are not resource-constrained anymore, thus the optimal solution is the MPM-schedule.

RS_R	0.20	0 .50	0.70	1.0
μ_{CPU}	1551.52	247.83	45.60	0.03

Table 13: Effects of the Resource Strength RS_R on Solution Times

In order to get even more insight into the effects of the parameters on the solution time, we have chosen the combination C=1.5, RF=0.5 and RS=0.5 for which an average solution time of 23.59 seconds was needed. Using a ceteris paribus design we changed just one parameter at a time and generated again 10 instances for each parameter level remaining w.r.t. Tables 8 and 9.

The effect of the number of renewable resources can be seen in Table 14. It is quite intuitive that an increasing number of constrained resources complicates the problem.

$\|R\|$	1	2	3	4	5	6
μ_{CPU}	0.09	1.10	4.29	23.59	138.51	406.15

Table 14: Effects of the Number of Resources $|R|$ on Solution Times

The effects of the number of start activities is depicted in Table 15. Increasing the number of start activities, keeping the number of jobs and precedence relations constant, generally results in more parallelism of the network, which makes the problem harder to solve.

S_1	1	2	3	4	5	6
μ_{CPU}	2.75	8.74	23.59	33.70	90.97	134.29

Table 15: Effects of the Number of Start Activities S_1 on Solution Times

Reasoned by the strong impact of the resource strength on solution time, indicated in the full factorial design study, a more thoroughly study on the RS_R has been performed. Table 16 shows the results of varying RS_R from 0 to 1 in steps of 0.1. The average solution time continuously increases with decreasing RS_R. The hardest problems are the ones where the minimal resource availability is provided. This relationship between hardness of the problem and resource scarcity deviates from the function conjectured by Elmaghraby and Herroelen (cf. [22]) and the computational study presented by Alvarez-Valdes/Tamarit (cf. [1]).

Finally the effect of a growing number of jobs is outlined in Table 17. Since it is well known that the problem is NP-complete with respect to the number of activities (cf. [27]), it is not surprising that solution times

RS_R	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
μ_{CPU}	3203	2545	1177	739	573	23.59	16.15	1.62	0.47	0.09	0.04

Table 16: Effects of the Resource Strength RS_R on Solution Times

grow rapidly with the number of jobs.

J	10	20	30	40
μ_{CPU}	0.06	0.32	23.59	942.09

Table 17: Effects of the Number of Jobs J on Solution Times

To sum it all up, even the single-mode case is less tractable than suggested by previously published work based on the Patterson test data.

5.2 Multi-Mode Case

Once more for makespan minimization problems we conjecture that the effects of the complexity, the number of constrained resources, the number of start activities and the number of jobs are about the same for the single- and the multi-mode case. Therefore we concentrated on the mutually effects of the resource factor and the resource strength for renewable and nonrenewable resources. Again we have utilized a full factorial design with the constant and varying parameter levels as given in Table 18 and 19, respectively. With 10 instances for each level combination of the varying parameters we generated $4 \cdot 4 \cdot 2 \cdot 2 \cdot 10 = 640$ problems. Each problem has been solved with the state of the art solution procedure of Patterson et al. (cf. [41]). It is a branch & bound based enumeration algorithm of the backtracking variety. Computational results are given in Patterson et al. (cf. [42]). There, 91 instances have been generated with characteristics similar to the ones of the 110 instances by Patterson. The number of jobs ranged between 10 and 500, where 75 instances had up to 30 jobs. The solution procedure has been coded in Fortran and implemented on an IBM 4381 mainframe computer. For an imposed time limit of 1 (10) minutes 30 (33) of the problems with up to 50 jobs could be solved to optimality. The preponderance of these problems ranged between ten and thirty jobs.

Since the original solution procedure was not available to us, we recoded it in C. Our code has been implemented on an IBM RS/6000 550 workstation, which is approximately 5 to 6 times faster than the IBM 4381 mainframe and about 50 times faster than the IBM PS/2 55sx. Because, as already pointed out in Section 4, we could not guarantee feasibility, only 536 of the 640 problems had a feasible solution. The average time to

	J	M_j	d_j	$\|R\|$	U_R	Q_R	$\|N\|$	U_N	Q_N	S_1	S_j	P_J	P_j
min	10	3	1	2	1	1	2	1	1	3	1	3	1
max	10	3	10	2	10	2	2	10	2	3	3	3	3

Table 18: Constant Parameter Levels for the Multi-Mode Instances under Full Factorial Design

Parameter	Levels			
RS	0.5	1.0		
RS	0.2	0.5	0.7	1.0

Table 19: Variable Parameter Levels for Multi-Mode Instances under Full Factorial Design

find and verify the optimal solution was 74.31 seconds. The minimum and maximum time was less than 0.5 seconds and 2016.25 seconds, respectively. Table 20 gives the frequency distribution of the solution times.

Range	[0,0.1]	(0.1,1]	(1,5]	(5,10]	(10,25]	(25,50]	(50,100]	(100,250]	≥250
Instances	142	40	76	50	62	38	31	46	51

Table 20: Frequency Distribution of Solution Times for the Multi-Mode Instances

In Table 21 the effects of varying resource factors is documented. With an increasing resource factor problems become harder. Solution times are far more sensitive to RF_N (factor 15) than to RF_R (factor 1.5).
The effects of the resource strength can be seen in Table 22. As the nonrenewable resources become scarce, problems turn to be much more difficult. Amazingly this does not hold for the renewable resources in general. The bottom line of Table 22 shows that the reverse is true; problems become harder to solve with increasing availability. If one recalls the results of the single-mode case, this is quite unexpected. But a more thorough study of Table 22 provides an explanation. In the case of sufficient nonrenewable resources, i.e. $RS_N \geq 0.7$, solution times increase with decreasing availability of renewable resources. But with small amounts of nonrenewable resources ($RS_R \leq 0.5$) the effect reverses. Due to the strong impact of RS_N the mean solution time only shows the tendency for scarce nonrenewable resources.
To sum it all up, we could not reproduce the promising results provided by Patterson et al. (cf. [42]) for the multi-mode case. Moreover, multi-mode instances in general are tractable only for a very restricted number of jobs. Thus additional work has to be done to speed up convergence.

		RF_R		
		0.5	1.0	
RF_N	0.5	6.92	9.24	8.10
	1.0	105.55	142.44	124.85
		62.14	85.70	74.31

Table 21: Effects of Varying Resource Factor

		RS_R				
		0.2	0.5	0.7	1.0	
RS_N	0.2	267.86	281.51	441.03	443.08	363.13
	0.5	38.69	46.80	58.71	101.52	62.47
	0.7	15.14	14.97	12.84	11.25	13.53
	1.0	12.66	3.27	0.48	0.06	3.57
		54.96	59.20	84.70	95.53	74.31

Table 22: Effects of Varying Resource Strength

6 Conclusions

PROGEN, a project generator for a broad class of precedence- and resource-constrained scheduling problems, which utilizes well-known and new summary measures, has been presented. Benchmark instances for the single- and the multi-mode case of project scheduling have been produced and solved with the state of the art B&B-procedures.

The results show the strong impact of the proposed parameters, furthermore very hard and very easy instances can be discriminated. In general, the promising results of previously published studies do not hold true; i.e. even very small problem instances still remain untractable with the optimal state of the art algorithms.

The availability of the generator as well as the 1216 instances used in the computational study provide a tool for the evaluation of algorithms within the project scheduling environment. Due to the versatility of the generator it can be used in related areas, e.g. single- and multiple-machine scheduling.

Acknowledgement: We thank Erik Demeulemeester, Katholieke Universiteit Leuven, for providing us with the code of his algorithm.

Appendix

A Functional Description of PROGEN

PROGEN has been coded in Borland Turbo Pascal 6.0. The code consists of the following eight units (cf. Table 23) with the corresponding tasks. All units except TYPEDECL have already been compiled and are available in the Turbo Pascal unit format (TPU). The code of TYPEDECL and PROGEN is amenable, so that the users can adjust the size of arrays to their specific needs. After adjustment TYPEDECL has to be compiled and all eight units have to be linked under Turbo Pascal 6.0 with the BUILD command. User who do not wish to change the size of arrays can use the readily available execution file of PROGEN.

PROGEN	:	main program.
TYPEDECL	:	definition of constants, types and variables (data structures).
NETWGEN	:	generation of the network.
REQGEN	:	generation of the resource request and level of demand.
AVAILGEN	:	generation of the resource availability.
INOUT	:	read and write routines.
UTILITY	:	support functions, e.g. the random number generator of Schrage [49].

Table 23: Units of PROGEN

When starting PROGEN one needs a file with the parameter settings, henceforth refered to as basedata-file. The basedata-file has always the suffix BAS. In Table 24 an example of such a basedata-file is depicted. The input relates to the parameters as presented in sections 3 and 4. Starting PROGEN one gets the menu shown in Table 25. In option "1" one has to choose a basedata-file, e.g. EXPL.BAS. The basedata-file is checked for existence on the actual subdirectory. Option "2" allows one to define a seed for the implemented random number generator. By default the random number generator of Turbo Pascal will be invoked once to generate the seed for the congruence-generator. The default value for the number of instances is 10. If a different number of instances is required, one can use option "3" for an adjustment. All adjustments are displayed in the upper right part of the menu. With option "4" the instance generator is started. It will create the predescribed number of instances. The instances have the same name as the basedata-file, but with the suffix DAT. They are labeled consecutively, e.g. one will get the files EXPL1.DAT to EXPL10.DAT. The warning and error messages of the generated instances will be written in a separate file, which also has the name of the basedata-file and the suffix ERR, e.g. EXPL.ERR. The possible error messages are shown in Table 26. They can be divided in four classes. Messages about wrong input (11-22), messages about

the process of generation (1,2,29), messages about the nontolerated deviation of parameters (3,4,23-28) and serious errors, which will lead to the interruption of the generation process (1000-1002). An example for an instance file and the corresponding error file is displayed in Tables 27 and 28, respectively. ERROR 1 and ERROR 1001 should not occur, if it does, please send input file and seed to the authors. In order to avoid the user from unintentional erasing instance files one cannot generate problems from a basedata-file, if an error file with the same name already exists in the actual subdirectory. Therefore those instances have to be erased before restarting the generation. In case of any problems please contact one of the authors.

```
SAMPLEFILE BASEDATA
PROJEKTS
  NrOfPro           :  1      & number of projects
  MinJob            :  8      & minimal number of jobs per project
  MaxJob            :  8      & maximal number of jobs per project
  MaxRelDate        :  0      & maximal release date
  DueDateFactor     : 0.0     & maximal due date
MODES
  MinMode           :  1      & minimal number of modes
  MaxMode           :  2      & maximal number of modes
  MinDur            :  1      & minimal duration
  MaxDur            :  10     & maximal duration
NETWORK
  MinOutSource      :  1      & minimal number of start activities per project
  MaxOutSource      :  3      & maximal number of start activities per project
  MaxOut            :  3      & maximal number of successor per job
  MinInSink         :  1      & minimal number of finish activities
  MaxInSink         :  2      & maximal number of finish activities
  MaxIn             :  3      & maximal number of predecessors
  Complexity        :  1.5    & complexity of network
RESSOURCEREQUEST/AVAILABILITY
  Rmin              :  2      & minimal number of renewable resources
  Rmax              :  2      & maximal number of renewable resources
  RminDemand        :  1      & minimal (per period) demand
  RmaxDemand        : 10      & maximal (per period) demand
  RRMin             :  1      & minimal number of resources requested
  RRMax             :  2      & maximal number of resources requested
  RRF               :  0.5    & resource factor
  RRS               :  0.2    & resource strength
  Number R-Func.    :  2
    p1              :  0.0    & probability to choose a constant function
    p2              :  1.0    & probability to choose a decreasing function
  Nmin              :  2      & cf. renewable resources
  Nmax              :  2
  NminDemand        :  1
  NmaxDemand        :  10
  NRMin             :  1
  NRMax             :  2
  NRF               :  1.0
  NRS               :  0.7
  Number N-Func.    :  2
    p1              :  0.0
    p2              :  1.0
```

```
  Dmin                : 0     & cf. renewable resources
  Dmax                : 0
  DminDemand          : 0
  DmaxDemand          : 0
  DRMin               : 0
  DRMax               : 0
  DRF                 : 0.0
  DRST                : 0.0
  DRSP                : 0.0
  Number D-Func.      : 2
    p1                : 1.0
    p2                : 0.0
LIMIT OF ITERATIONS
  Tolerance Network   : 0.05  & tolerated complexity deviation
  Tolerance RF        : 0.05  & tolerated resource factor deviation

  MaxTrials           : 200   & maximal number of trials

FORMAT OF BASE FILE
  - a colon has to be followed by a value
  - only spaces are allowed between colon and value
  - a comment is allowed to follow a value
  - comments are allowed if there is no colon in
  - value and comment have to be seperated by space
  - value is integer with the exception of
  -> due date factor   -> complexity             -> resource factor
  -> resource strength -> function probabilities -> tolerances
```

Table 24: Parameter Settings in the Basedata-File

```
======================================================================
Project Generator PROGEN (Version 2.0)
======================================================================

file basedata       : no basefile
initial value       : randomly
number of instances :       10

1 - basedata
2 - initial value
3 - number of instances
4 - generate
5 - end program

-->
```

Table 25: Menue of Progen

```
ERROR   1: Predecessor could not be determined.
ERROR   2: Successor could not be determined.
ERROR   3: Complexity could not be achieved (low).
ERROR   4: Complexity could not be achieved (high).
ERROR  11: max # req. resources > # resources for type R; -> max# := #.
ERROR  12: max # req. resources > # resources for type D; -> max# := #.
ERROR  13: max # req. resources > # resources for type N; -> max# := #.
ERROR  14: min # req. resources > max # for type R; -> min # := max #.
ERROR  15: min # req. resources > max # for type D; -> min # := max #.
ERROR  16: min # req. resources > max # for type N; -> min # := max #.
ERROR  17: RF for R can't be achieved; min # req. resources too large.
ERROR  18: RF for D can't be achieved; min # req. resources too large.
ERROR  19: RF for N can't be achieved; min # req. resources too large.
ERROR  20: RF for R can't be achieved; max # req. resources too small.
ERROR  21: RF for D can't be achieved; max # req. resources too small.
ERROR  22: RF for N can't be achieved; max # req. resources too small.
ERROR  23: Obtained RF falls short the tolerated range for R.
ERROR  24: Obtained RF falls short the tolerated range for D.
ERROR  25: Obtained RF falls short the tolerated range for N.
ERROR  26: Obtained RF exceeds the tolerated range for R.
ERROR  27: Obtained RF exceeds the tolerated range for D.
ERROR  28: Obtained RF exceeds the tolerated range for N.
ERROR  29: More than 1 trial was used to produce a job with non dominated modes.

ERROR1000: Network generation without success.
ERROR1001: Redundant arcs in network.
ERROR1002: Non dominated modes for a job could'nt be produced within maxtrials.
```

Table 26: Error Messages

```
******************************************************************************
file with basedata            : expl.bas
initial value random generator: 530450642
******************************************************************************
projects                      :  1
jobs (incl. supersource/sink ):  10
horizon                       :  47
RESOURCES
  - renewable                 :  2   R
  - nonrenewable              :  2   N
  - doubly constrained        :  0   D
******************************************************************************
PROJECT INFORMATION:
pronr.  #jobs rel.date duedate tardcost  CPM-Time
    1      8      0       20        2       20
```

```
******************************************************************************
PRECEDENCE RELATIONS:
jobnr.    #modes  #successors   successors
   1        1          2           2   3
   2        2          3           6   7   9
   3        2          3           4   5   9
   4        1          1           7
   5        2          2           6   7
   6        2          1           8
   7        1          1           8
   8        1          1          10
   9        1          1          10
  10        1          0
******************************************************************************
REQUESTS/DURATIONS:
jobnr. mode duration  R 1  R 2  N 1  N 2
------------------------------------------------------------------------------
  1      1     0       0    0    0    0
  2      1     6       7    0    2    5
         2     9       7    0    1    3
  3      1     1       0    7    8    7
         2     3       0    5    8    4
  4      1     3       3    0    9    3
  5      1     7       5    0    8   10
         2     7       0    4    5    6
  6      1     1       0    6    8    8
         2     9       0    4    2    8
  7      1     2       2    0    4    3
  8      1    10       3    0    1    5
  9      1     4       0   10    6    5
 10      1     0       0    0    0    0
******************************************************************************
RESOURCEAVAILABILITIES:
  R 1  R 2  N 1  N 2
    9   11   43   43
******************************************************************************
```

Table 27: Example Instance File

```
------------------------------------------------------------------------------
sample file -->expl1.DAT
------------------------------------------------------------------------------
ERROR   2: Successor could not be determined
ERROR   3: Complexity could not be achieved (low)
ERROR   2: Successor could not be determined
ERROR   3: Complexity could not be achieved (low)
ERROR  29: More than 1 trial was used to produce a job with non dominated modes
------------------------------------------------------------------------------
```

Table 28: Example File Error Messages

References

[1] ALVAREZ-VALDES, R. AND J.M. TAMARIT (1989): Heuristic algorithms for resource-constrained project scheduling: A review and an empirical analysis. In: Slowinski, R. and J. Weglarz (Eds.): Advances in project scheduling. Elsevier, Amsterdam, pp. 113-134.

[2] BALAS, E. (1971): Project scheduling with resource constraints. In: Beale, E.M.L. (Ed.): Applications of mathematical programming techniques. The English Universities Press, London, pp. 187-200.

[3] BARTUSCH, M.; R.H. MÖHRING AND F.J. RADERMACHER (1988): Scheduling project networks with resource constraints and time windows. Annals of Operations Research, Vol. 16, pp. 201-240.

[4] BELL, C.E. AND J. HAN (1991): A new heuristic solution method in resource-constrained project scheduling. Naval Research Logistics, Vol. 38, pp. 315-331.

[5] BELL, C.E. AND K. PARK (1990): Solving resource-constrained project scheduling problems by A* search. Naval Research Logistics, Vol. 37, pp. 61-84.

[6] BOCK, D.B. AND J.H. PATTERSON (1990): A comparison of due date setting, resource assignment, and job preemption heuristics for the multiproject scheduling problem. Decision Sciences, Vol. 21, pp. 387-402.

[7] BOCTOR, F.F. (1992): Heuristics for scheduling projects with resource restrictions and several resource-duration modes. Working Paper, Pavillon des Sciences de l'Administration, Universite Laval, Quebec, Canada.

[8] BOWMAN, E.H. (1959): The schedule-sequencing problem. Operations Research, Vol. 7, pp. 621-624.

[9] CHRISTOFIDES, N.; R. ALVAREZ-VALDES AND J.M. TAMARIT (1987): Project scheduling with resource constraints: A branch and bound approach. European Journal of Operational Research, Vol. 29, pp. 262-273.

[10] COOPER, D.F. (1976): Heuristics for scheduling resource-constrained projects: An experimental investigation. Management Science, Vol. 22, pp. 1186-1194.

[11] DAVIES, E.M. (1973): An experimental investigation of resource allocation in multi activity projects. Operational Research Quarterly (since 1978: Journal of the Operational Research Society), Vol. 24, pp. 587-591.

[12] DAVIS, E.W. (1968): An exact algorithm for the multiple constrained-resource project scheduling problem. PhD Dissertation, Yale University, New Haven, USA.

[13] DAVIS, E.W. (1975): Project network summary measures constrained-resource scheduling. AIIE Transactions (since 1985: IIE Transactions), Vol. 7, pp. 132-142.

[14] DAVIS, E.W. AND G.E. HEIDORN (1971): An algorithm for optimal project scheduling under multiple resource constraints. Management Science, Vol. 17, pp. B803-B816.

[15] DAVIS, E.W. AND J.H. PATTERSON (1975): A comparison of heuristic and optimum solutions in resource-constrained project scheduling. Management Science, Vol. 21, pp. 944-955.

[16] DECKRO, R.F. AND J.E. HEBERT (1989): Resource constrained project crashing. OMEGA, Vol. 17, pp. 69-79.

[17] DEMEULEMEESTER, E. (1992): Optimal algorithms for various classes of multiple resource-constrained project scheduling problems. PhD Dissertation, Katholieke Universiteit Leuven, Belgium.

[18] DEMEULEMEESTER, E. AND W. HERROELEN (1992): A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. Management Science, to appear.

[19] DREXL, A. (1991): Scheduling of project networks by job assignment. Management Science, Vol. 37, pp. 1590-1602.

[20] DREXL, A. AND J. GRÜNEWALD (1992): Nonpreemptive multi-mode resource-constrained project scheduling. IIE Transactions , to appear.

[21] ELMAGHRABY, S.E. (1977): Activity networks: Project planning and control by network models. Wiley, New York.

[22] ELMAGHRABY, S.E. AND W.S. HERROELEN (1980): On the measurement of complexity in activity networks. European Journal of Operational Research, Vol. 5, pp. 223-234.

[23] HERROELEN, W.; E. DEMEULEMEESTER AND B.DODIN (1989): The generation of strongly-random activity networks. Working Paper, Department of Applied Economic Sciences, Katholieke Universiteit Leuven, Belgium.

[24] JACKSON, H.F.; P.T. BOGGS; S.G. NASH AND S. POWELL (1991): Guidelines for reporting results of computational experiments. Report of the ad hoc committee. Mathematical Programming, Vol. 49, pp. 413-425.

[25] JOHNSON, T.J.R. (1967): An algorithm for the resource-constrained project scheduling problem. PhD Dissertation, Massachusets Institute of Technology, USA.

[26] KAIMANN, R.A. (1974): Coefficients of network complexity. Management Science, Vol. 21, pp. 172-177.

[27] KARP, R.M. (1972): Reducibility among combinatorial problems. In: Miller, R.E. and J.W. Thatcher (Eds.): Complexity of computer applications. Plenum Press, New York, pp. 85-104.

[28] KIM, S. AND R.C. LEACHMAN (1990): A hierarchical approach to multi-resource multi-project scheduling with explicit lateness costs. IIE Transactions, to appear.

[29] KURTULUS, I.S. (1983): Multi-project scheduling: Analysis of project performance. Working Paper, School of Business, VCU, Richmond, USA.

[30] KURTULUS, I.S. AND E.W. DAVIS (1982): Multi-project scheduling: Categorization of heuristic rules performance. Management Science, Vol. 28, pp. 161-172.

[31] KURTULUS, I.S. AND S.C. NARULA (1985): Multi-project scheduling: Analysis of project performance. IIE Transactions, Vol. 17, pp. 58-66.

[32] LAWRENCE, S.R. AND T.E. MORTON (1991): Resource-constrained multi-project scheduling with tardy costs: Comparing myopic, bottleneck, and resource pricing heuristics. Working Paper, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, USA.

[33] MASON, A.T. AND C.L. MOODIE (1971): A branch and bound algorithm for minimizing cost in project scheduling. Management Science, Vol. 18, pp. B158-B173.

[34] MASTOR, A.A. (1970): An experimental investigation and comparative evaluation of production line balancing techniques. Management Science, Vol. 16, pp. 728-746.

[35] NEUMANN, K. (1975): Operations Research Verfahren, Bd. 3. Hanser, München-Wien.

[36] PASCOE, T.L. (1966): Allocation of resources C.P.M. Revue Francaise Recherche Operationelle, No. 38, pp. 31-38.

[37] PATTERSON, J.H. (1976): Project scheduling: The effects of problem structure on heuristic performance. Naval Research Logistics Quarterly (since 1987: Naval Research Logistics), Vol. 23, pp. 95-123.

[38] PATTERSON, J.H. (1984): A comparison of exact approaches for solving the multiple constrained resource, project scheduling problem. Management Science, Vol. 30, pp. 854-867.

[39] PATTERSON, J.H. AND W.D. HUBER (1974): A horizon-varying, zero-one approach to project scheduling. Management Science, Vol. 20, pp. 990-998.

[40] PATTERSON, J.H. AND G.W. ROTH (1976): Scheduling a project under multiple resource constraints: A zero-one programming approach. AIIE Transactions (since 1985: IIE Transactions), Vol. 8, pp. 449-455.

[41] PATTERSON, J.; R. SLOWINSKI; B. TALBOT AND J. WEGLARZ (1989): An algorithm for a general class of precedence and resource constrained scheduling problems. In: Slowinski, R. and J. Weglarz (Eds.): Advances in project scheduling. Elsevier, Amsterdam, pp. 3-28.

[42] PATTERSON, J.; R. SLOWINSKI; B. TALBOT AND J. WEGLARZ (1990): Computational experience with a backtracking algorithm for solving a general class of precedence and resource-constrained scheduling problems. European Journal of Operational Research, Vol. 49, pp. 68-79.

[43] PRITSKER, A.A.B.; W.D. WATTERS AND P.M. WOLFE (1969): Multiproject scheduling with limited resources: A zero-one programming approach. Management Science, Vol. 16, pp. 93-108.

[44] RADERMACHER, F.J.(1985/6): Scheduling of project networks. Annals of Operations Research, Vol. 4, pp. 227-252.

[45] RUSSELL, A.H. (1970): Cash flows in networks. Management Science, Vol. 16, pp. 357-373.

[46] RUSSELL, R.A. (1986): A comparison of heuristics for scheduling projects with cash flows and resource restrictions. Management Science, Vol. 32, pp. 1291-1300.

[47] SAMPSON, S.E. AND E.N. WEISS (1992): Local search techniques for the resource constrained project scheduling problem. Research Report, The Darden School, University of Virginia, USA.

[48] SCHRAGE, L. (1971): Solving resource-constrained network problems by implicit enumeration - non-preemptive case. Operations Research, Vol. 18, pp. 263-278.

[49] SCHRAGE, L. (1979): A more portable fortran random number generator. ACM Transactions on Mathematical Software, Vol. 5, pp.132-138.

[50] SLOWINSKI, R. (1978): A node ordering heuristic for network scheduling under multiple resource constraints. Foundations of Control Engineering, Vol. 3, pp. 19-27.

[51] SLOWINSKI, R. (1980): Two approaches to problems of resource allocation among project activities: A comparative study. Journal of the Operational Research Society, Vol. 31, pp. 711-723.

[52] SLOWINSKI, R. (1981): Multiobjective network scheduling with efficient use of renewable and nonrenewable resources. European Journal of Operational Research, Vol. 7, pp. 265-273.

[53] SMITH-DANIELS, D.E. AND V.L. SMITH-DANIELS (1987): Optimal project scheduling with materials ordering. IIE Transactions, Vol. 19, pp. 122-129.

[54] STINSON, J.P. (1976): A branch and bound algorithm for a general class of multiple resource-constrained scheduling problems. PhD Dissertation, Graduate School of Business Administration, University of North Carolina, USA.

[55] STINSON, J.P.; E.W. DAVIS AND B.M. KHUMAWALA (1978): Multiple resource-constrained scheduling using branch and bound. AIIE Transactions (since 1985: IIE Transactions), Vol. 10, pp. 252-259.

[56] TALBOT, F.B. (1980): Project scheduling with resource-duration interactions: The nonpreemptive case. Working Paper, The Graduate School of Business Administration, University of Michigan, USA.

[57] TALBOT, F.B. (1982): Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case. Management Science, Vol. 28, pp. 1197-1210.

[58] TALBOT, F.B. AND J.H. PATTERSON (1978): An efficient integer programming algorithm with network cuts for solving resource-constrained scheduling problems. Management Science, Vol. 24, pp. 1163-1174.

[59] THESEN, A. (1976): Heuristic scheduling of activities under resource and precedence restrictions. Management Science, Vol. 23, pp. 412-422.

[60] THESEN, A. (1977): Measures of the restrictiveness of project networks. Networks, Vol. 7, pp. 193-208.

[61] ULUSOY, G. AND L. ÖZDAMAR (1989): Heuristic performance and network/resource characteristics in resource-constrained project scheduling. Journal of the Operational Research Society, Vol. 40, pp. 1145-1152.

[62] WEGLARZ, J. (1979): Project scheduling with discrete and continuous resources. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 9, pp. 644-650.

[63] WEGLARZ, J. (1980): On certain models of resource allocation problems. Kybernetics, Vol. 9, pp. 61-66.

[64] YAU, C. AND E. RITCHIE (1988): A linear model for estimating project resource levels and target completion times. Journal of the Operational Research Society, Vol. 39, pp. 855-866.