Characterization and Generation of a General Class of Resource-constrained Project Scheduling Problems

Rainer Kolisch • Arno Sprecher • Andreas Drexl
Institut für Betriebswirtschaftslehre, Christian-Albrechts-Universität zu Kiel, Olshausenstraße 40, 24098 Kiel, Germany

This paper addresses the issue of how to generate problem instances of controlled difficulty. It focuses on precedence- and resource-constrained (project) scheduling problems, but similar ideas may be applied to other network optimization problems It describes a network construction procedure that takes into account a) constraints on the network topology, b) a resource factor that reflects the density of the coefficient matrix, and c) a resource strength, which measures the availability of resources The strong impact of the chosen parametric characterization of the problems is shown via an in depth computational study. 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 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 large amount of computation time.

(*Project Scheduling; Precedence- and Resource-Constraints; Nonpreemptive Case; Single-Mode, Multi-Mode; Branch and Bound Methods; Project Generator; Benchmark Instances*)

1. Introduction

From the beginning of resource-constrained project scheduling research, rapid progress regarding models and methods has been documented in the literature (eg Demeulemeester and Herroelen 1992; Patterson et al 1989; Pritsker, Watters and Wolfe 1969; Stinson, Davis and Khumawala 1978; and Talbot 1982) But at the same time very little research concerned with the systematic generation of benchmark instances has been published To the best of our knowledge, the only paper explicitly dealing with an instance generator for project scheduling problems has been presented by Demeulemeester et al (1993) There, a generator for the random generation of activity-on-arc (AOA) networks is presented Two disadvantages should be mentioned. (i) Beside the number of nodes and arcs, respectively, no further network characteristics can be specified Hence, special network structures cannot be generated intentionally; (ii) The resource demand and availability generation (cf §4) is only performed in accordance with general distributions without making use of a specified set of characteristics

Besides Demeulemeester et al (1993), numerous authors have documented their way of obtaining instances as a vehicle for their numerical experiments (cf Alvarez-Valdes and Tamarit 1989; Drexl 1991, Kurtulus and Davis 1982, Pascoe 1966; Patterson 1984; and Patterson et al 1990) Thereby, different project characteristics have been suggested Worth mentioning are the papers of Pascoe (cf 1966) Kurtulus and Davis (cf. 1982) as

0025-1909/95/4110/1693$01 25
Copyright © 1995 Institute for Operations Research and the Management Sciences

Copyright © 2001 All Rights Reserved

well as Cooper (cf. 1976) But, as to be discussed in §4 all of them encountered difficulties with the important measure of resource scarceness

Today, 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 problem on 110 instances (cf Patterson 1984). Amongst others, these instances have been used by Bell and Han (1991), Demeulemeester and Herroelen (1992), and Sampson and Weiss (1993) and therefore became a quasi standard. Nevertheless, they have several weaknesses. First, as a collection of problems from different sources, the problems are not generated by using a controlled design of specified parameters. Second, only the single-mode case and makespan minimization is taken into consideration Third, recent advances (cf Demeulemeester and Herroelen 1992) in the development of exact single-mode procedures have demonstrated that the Patterson-set is solvable within an average CPU-time of 0.76 seconds and a maximum CPU-time of less than 14 0 seconds on an IBM PS/2 Model 55sx (80386 sx processor, 15 Mhz clockpulse). Since there are instances (with the same number of activities), namely those with a high resource factor and a low resource strength, which are much more difficult to solve, the Patterson-set can no longer be considered as a benchmark

Therefore, the intention of the paper is twofold: First, we present an instance generator for a broad class of project scheduling problems which utilizes several parameters, i e. project characteristics 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 specified parameters 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 §2 we give a formal description of the model. The employed parameters and their realization within the project generator are dealt with in §§3 and 4 The effect of the parameters used in the computational study of the single- and multi-mode case, respectively, is outlined in §5 Conclusions can be found in §6

2. Notation and Model Description

We consider a project which consists of J partially ordered jobs, where $j = 1$ ($j = J$) is the unique dummy source (sink) 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 (AON) network.

We distinguish three categories of (scarce) resources (cf. Blazewicz et al 1986): The set R of renewable resources, the set N of nonrenewable resources, and finally the set D of doubly constrained resources A resource $r \in R$ has a constant period capacity of K_r and a resource $r \in N$ has an overall capacity of K_r units. Doubly constrained resources $r \in D$ are limited with respect to period capacity and total capacity. 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} units of the renewable resource r each period it is in process and consumes k_{jmr} units of the nonrenewable resource r Obviously, a doubly constrained resource can be incorporated by defining a matching renewable and nonrenewable resource, respectively.

The constraints are given in Table 1 For modelling purposes we use binary variables x_{jmt}, $j = 1, \ldots, J$, $m = 1, \ldots, M_j$, $t = EF_j, \ldots, LF_j$, as proposed in Pritsker, Watters and Wolfe (1969) x_{jmt} is equal to one if job j is performed in mode m and completed in period t, and zero otherwise. (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 traditional forward and backward recursion by using $\bar{T}$ as an upper bound of the makespan and the modes of shortest duration. Actually, $\bar{T}$ may be defined as the sum of the maximal durations. Precedence relations between related jobs are maintained by (2). (3) secures feasibility with respect to renewable resources Finally, (4) limits the consumption of the nonrenewable resources to their availability. This formulation embodies a wide range of precedence- and resource-constrained scheduling problems, especially

Copyright © 2001 All Rights Reserved

Table 1 Constraints

$\sum_{m=1}^{M_j} \sum_{t=EF_j}^{LF_j} x_{jmt} = 1$	$j = 1, \ldots, J$	(1)
$\sum_{m=1}^{M_h} \sum_{t=EF_h}^{LF_h} t x_{hmt} \leq \sum_{m=1}^{M_j} \sum_{t=EF_j}^{LF_j} (t - d_{jm}) x_{jmt}$	$j = 2, \ldots, J, h \in P_j$	(2)
$\sum_{j=1}^{J} \sum_{m=1}^{M_j} k_{jmr} \sum_{q=t}^{t+d_{jm}-1} x_{jmq} \leq K_r$	$r \in R, t = 1, \ldots, T$	(3)
$\sum_{j=1}^{J} \sum_{m=1}^{M_j} k_{jmr} \sum_{t=EF_j}^{LF_j} x_{jmt} \leq K_r$	$r \in N$	(4)
$x_{jmt} \in \{0, 1\}$	$j = 1, \ldots, J, m = 1, \ldots, M_j,$ $t = EF_j, \ldots, LF_j$	(5)

the single- and the multi-mode version 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.

A common objective w.r.t. (1)–(5) is the minimization of the makespan, i.e.,

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

Note that the main emphasis of the paper is on the generation of the constraints (1)–(5). In addition, it is easy to incorporate other (regular) objective functions as, e.g., the minimization of the weighted project delay. The details are left to the reader (and user of ProGen).

3. Project Generation

We use the functions "round" and "trunc," where the former rounds a real argument to an integer and the latter truncates the decimal fraction of a given real. Moreover, the random function "rand" ("$\overline{\text{rand}}$") is defined by drawing a uniformly distributed integer (real) out of a specified interval.

The (pseudo) random numbers are constructed by transforming [0, 1) uniformly distributed random numbers, which are calculated via the congruence-generator developed by Lehmer using the constants and implementation as given in Schrage (1979).

3.1. Base Data Generation

The input of the base data generation is given by the minimal (maximal) number of jobs $J^{\min}$ ($J^{\max}$), the minimal (maximal) number of modes $M^{\min}$ ($M^{\max}$) and the minimal (maximal) duration $d^{\min}$ ($d^{\max}$). The output is determined by applying "rand" to the related interval.

3.2. Network Generation

In §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 a simple implication of the definition of a network: 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 use 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 (h, j) is redundant, if it is an element of the transitive closure $\bar{N}^+$ of $\bar{N} = (V, A \setminus \{(h, j)\})$. If within the construction process of the network an arc (h, j) is chosen for adding it to the current graph it might happen that the (intermediately) derived graph bears redundant arcs. To avoid this four cases of redundancy which might occur have to be considered (cf. Kolisch, Sprecher and Drexl 1992).

For a given cardinality of the set of nodes the minimal and maximal number of non-redundant arcs of a network are given in the following theorem.

THEOREM 1. *Let $N = (V, A)$ be a network with $|V| = n$.*

Copyright © 2001 All Rights Reserved

(a) *Since a network is connected, the minimal number of nonredundant arcs $A^{\min}$ is given by $A^{\min} = n - 1$*

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

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

For the characterization of the network we use the minimal (maximal) number of start activities $S_1^{\min}$ ($S_1^{\max}$), the minimal (maximal) number of finish activities $P_J^{\min}$ ($P_J^{\max}$) as well as the maximal number of successor (predecessor) activities $S_j^{\max}$ ($P_j^{\max}$) an activity j, $j = 2, \ldots, J-2$, can have. Moreover, the network complexity C, i e the average number of nonredundant arcs per node (including the super-source and -sink), and the tolerated complexity deviation ϵ_{NET} is used

The network complexity has been introduced by Pascoe (cf 1966) for AOA networks and adopted by Davis (cf. 1975) for the AON representation.

The construction of the network is performed in four steps, a detailed description of which is given in Kolisch, Sprecher and Drexl (1992) 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 In both steps the jobs are considered in order of increasing job number. Finally, (in Step 4) further arcs are added until the complexity is reached During the whole procedure one has to take into account. First, to avoid redundancy, there must be no precedence relations within the start-activities and within the finish-activities, respectively Second, adding arcs in Step 3 or 4 must not produce redundant precedence relations Third, the limitation given by the maximal number of successors (predecessors) as well as the number of start and finish activities has to be taken into account.

In the following cases the generation procedure has to be restarted: First, 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 ActArcs is too high, that is, ActArcs $> J \cdot C \cdot (1 + \epsilon_{NET})$ Second, if in Step 3, due to the limited number of predecessors there is no successor of a job j available Third, if in Step 3 for a job j, there are only successors available, which lead to redundant precedence relations Fourth, 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 $\geq J \cdot C \cdot (1 - \epsilon_{NET})$.

By an appropriate reduction of the set of choosable predecessors and successors in the steps previously described a numerically labeled network is realized. Through adjustment of the input parameters special network structures, e g serial, general, and parallel structures as well as the network shapes described in Kurtulus and Davis (1982) 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 consumed To the first step we refer with *request generation* (§4 1.1) and to the latter we refer with *generation of demand level* (§4 1 2)

We consider a resource category $\tau \in \{R, N\}$. The number of resources of category τ is determined by $|\tau| := \text{rand}[|\tau|^{\min}, |\tau|^{\max}]$.

4.1.1. Requested Resources. For characterizational purposes we use a generalization of the resource factor (RF) which has been introduced by Pascoe (cf. 1966) for the single-mode case and later on been utilized in studies by Cooper (cf. 1976) and Alvarez-Valdes / Ta-

Copyright © 2001 All Rights Reserved

mann (cf 1989) For the single-mode case RF is calculated as follows:

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

The resource factor reflects the average portion of resources requested per job It is a measure of the density of the array k_{jr} 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 category dependent resource factor RF_τ, $\tau \in \{R, N\}$:

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

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 category, 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 $|N| = 0$ and $M_j = 1$, $j = 1, \ldots, J$. Table 2 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, iff $[j, m]$ requests resource r. The current resource factor (ARF) is then calculated as follows:

Table 2 Input Demand Generation

$\|\tau\|^{\min}(\|\tau\|^{\max})$	minimal (maximal) number of resources of category τ
$Q_\tau^{\min}(Q_\tau^{\max})$	minimal (maximal) number of resources of category τ used by a job-mode combination $[j, m]$
$U_\tau^{\min}(U_\tau^{\max})$	minimal (maximal) demand for a resource of category τ
RF_τ	resource factor of category τ
RS_τ	resource strength of category τ
ϵ_{RF}	tolerated resource factor deviation

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

The current number of resources $Q[j, m]$ requested by $[j, m]$ is obtained by

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

Finally, we get CT, the current set of choosable triplets, by

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

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 $Q_\tau^{\max}$. 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 $Q_\tau^{\min}$ is not reached, additional resources are selected randomly While, in Step 2, the current resource factor is less than the asserted one and in addition there are choosable triplets in CT, i.e $\mathrm{CT} \neq \emptyset$, the current resource factor is incremented by randomly drawing a triplet out of CT while taking into account $Q_\tau^{\max}$.

If after Step 2 the current resource factor declines more then tolerated, i.e.,

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

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 and the other one is decreasing with the (increasing) duration (time-resource tradeoff) For each resource $r \in \tau$ the function is determined in accordance with category dependent probabilities. If a duration independent level is obtained, 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. If the level is monotonically increasing with respect to

Copyright © 2001 All Rights Reserved

the duration, then for each job j two levels are drawn randomly out of the parameter specified interval. The lower one defines U^{low} and the higher one U^{high}.

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

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

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

$$I_k := [\text{round}(U^{\text{high}} - \Delta k), \\ \text{round}(U^{\text{high}} - \Delta(k-1))], \quad k = 1, \ldots, \bar{M}_j$$

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. An illustration of the demand generation is given in Kolisch, Sprecher and Drexl (1992).

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 $k_{jmr} \leq k_{j\bar{m}r}$ for all $r \in R \cup N$.

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

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

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

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

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. 1976) introduced the resource strength (RS), which is calculated as follows:

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

Later the RS has been utilized by Alvarez-Valdes/Tamarit (cf. 1989). There are three drawbacks of the proposed measure: First, the RS is not normalized to the interval [0, 1]. Second, a rather small RS does not guarantee a feasible solution. For example, for three jobs with $k_{jr} = 1$, 1 and 10, respectively, one has to adjust the resource strength to $\text{RS}_r \geq 2.5$ 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 feasibility 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 measure of resource scarceness which is applicable to all categories 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 $K_r := K_r^{\min} + \text{RS}_r(K_r^{\max} - K_r^{\min})$. Thus, with respect to one resource we will get the smallest feasible resource availability for $\text{RS}_r = 0$. For $\text{RS}_r = 1$ the amount of resources is approximately large enough to achieve the MPM-case.

For the nonrenewable resources r, $r \in N$, the minimal and maximal availabilities ($K_r^{\min}$ and $K_r^{\max}$) to complete the project can be calculated as the sum of the minimal and maximal consumptions of the jobs. For a given category dependent resource strength $\text{RS}_r \in [0, 1]$ the availability is then

Copyright © 2001 All Rights Reserved

$$K_r := K_r^{\min} + \text{round}(RS_r(K_r^{\max} - K_r^{\min})).$$

If the considered resource is renewable the minimal demand is

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

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

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

and the corresponding mode with shortest duration:

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

Given the precedence relations 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

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

and the available amount using the category dependent resource strength RS_r

$$K_r := K_r^{\min} + \text{round}(RS_r(K_r^{\max} - K_r^{\min})). \qquad (6)$$

By construction we can state the following:

REMARK 2 (a) If $|r| = 1$ and $RS_r = 0$, then the lowest resource feasible level with respect to r will be generated

(b) If $RS_r \ll 1$ and $M_j > 1$, then feasibility of the problem cannot be assured, because of mode coupling via resource-constraints.

5. Computational Results

The outlined procedures have been coded in Turbo Pascal. The microcomputer (IBM PS/2 Model 55sx) implementation generates ten projects with thirty activities within 45 seconds If difficult network structures (i e , with very low or high complexity) have to be generated, the effort slightly increases. Ten multimode problems with ten activities and three modes can be obtained within 12 seconds

5.1. Single-mode Case

Currently the most advanced exact procedure for solving single-mode makespan minimization problems seems to be the algorithm from Demeulemeester and Herroelen (cf. 1992) It is an implicit enumeration procedure of the B&B type with backtracking. It is coded in C and solves the 43 27-job problems out of the 110 Patterson instances in an average computation 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 3 and 4, respectively Obviously, we have $|N| = 0$ and a duration independent level of demand 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, where we imposed a time limit of 3,600 seconds on the CPU time

Table 3 Constant Parameter Levels for Single-mode Instances Under Full Factorial Design

	J	M_j	d_j	$\lvert R \rvert$	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 4 Variable 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

Copyright © 2001 All Rights Reserved

Table 5 **Frequency Distribution of Solution Times for Single-Mode Instances Under Full Factorial Design**

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

Our 480 instances have been solved in 461.25 seconds on the average. The minimal solution time turned out to be 0.0 seconds (which is actually less than 0.05 seconds), while the maximal solution time was the imposed limit of 3,600 seconds. Table 5 provides the frequency distribution of the solution times. Among the 65 very hard problems which needed more than 1,000 CPU-seconds were 52 for which an optimal solution could neither be found nor verified. We evaluated the effects of the different parameters by the use of mean value analysis and ANOVA.

The complexity C turns out to be marginally significant ($\alpha = 0.16$). Table 6 shows that increasing the complexity from 1.5 to 2.1 induces a reduction of the average solution time. The rationale is 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 easier to solve. The effect has already been mentioned by Alvarez-Valdes and Tamarit for heuristics (cf. 1989).

The resource factor RF_R is highly significant ($\alpha = 0.0001$). Increasing RF_R results in an incline of solution times (cf. Table 7). This contradicts the results of Alvarez-Valdes and 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 §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.

The resource strength RS_R is also highly significant ($\alpha = 0.0001$). From Table 8 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 the 120 instances with $RS_R = 0.2$, for 47 problems the optimum solution could not be found or verified. Problems with an RS_R of 1.0 are no longer resource-constrained, thus the optimal solution is the MPM-schedule. This relationship between hardness of the problem and resource scarcity deviates from the function conjectured by Elmaghraby and Herroelen (cf. 1980) and the computational study presented by Alvarez-Valdes and Tamarit (cf. 1989).

In order to get deeper insight into the effects of the parameters on the solution time, we have chosen the combination $C = 1.5$, $RF_R = 0.5$ and $RS_R = 0.5$ 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 again generated 10 instances for each parameter level remaining w.r.t. Tables 3 and 4.

The effect of the number of renewable resources is statistically insignificant ($\alpha = 0.369$). Nevertheless, Table 9 shows an increasing average solution time with an enlarged number of renewable resources. It is quite intuitive that an increasing number of constrained resources complicates the problem.

The effects of the number of start activities $|S_1|$ is depicted in Table 10. Increasing the number of start

Table 6 **Effects of Complexity C on Solution Times**

C	1.5	1.8	2.1
μ_{CPU}	674.89	477.80	231.18
σ_{CPU}	1327.29	1169.04	827.44

Table 7 **Effects of the Resource Factor RF_R on Solution Times**

RF_R	0.25	0.50	0.75	1.0
μ_{CPU}	0.54	128.35	787.96	928.30
σ_{CPU}	1.81	526.69	1432.74	1498.02

Copyright © 2001 All Rights Reserved

Table 8 **Effects of the Resource Strength RS_R on Solution Times**

RS_R	0.20	0.50	0.70	1.00
μ_{CPU}	1551.70	247.83	45.60	0.03
σ_{CPU}	1703.82	752.55	352.99	0.02

Table 10 **Effects of the Number of Start Activities $\|S_1\|$ on Solution Times**

$\|S_1\|$	1	2	3	4	5	6
μ_{CPU}	2.75	8.74	23.59	33.70	90.97	134.29
σ_{CPU}	4.69	17.98	30.38	87.83	174.15	317.11

activities $|S_1|$, keeping the number of jobs and precedence relations constant, generally results in more parallelism of the network, which makes the problem harder to solve. The effect does not show statistical significance ($\alpha = 0.334$).

Finally, the effect of a growing number of jobs is outlined in Table 11 Since it is well known that the problem is NP-complete with respect to the number of activities (cf Karp 1972), it is not surprising that solution times grow rapidly with the number of jobs ANOVA reveals J to be significant ($\alpha = 0.024$)

To sum it up, each project characteristic shows a monotone influence on the average CPU-time Thereby, the resource strength and the resource factor are highly significant The number of jobs and the complexity are marginally significant, while the number of resources as well as the number of start activities are insignificant It can be concluded that 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 parameter levels as given in Table 3, but with $J^{\min} = J^{\max} = 10$ and $M_j^{\min} = M_j^{\max} = 3$ instead of $J^{\min} = J^{\max} = 30$ and $M_j^{\min} = M_j^{\max} = 1$, respectively Moreover, we used $|N| = 2$, $U_N^{\min} = 1$, $U_N^{\max} = 10$, $Q_N^{\min} = 1$, $Q_N^{\max} = 2$ and a decreasing level of usage (consumption) The full factorial design was performed by the use of the variable levels of $RF_\tau = 0.5, 1.0$ and $RS_\tau = 0.2, 0.5, 0.7, 1.0$ for both categories of resources 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. 1989) It is a B&B based enumeration algorithm of the backtracking variety. Computational results are given in Patterson et al (cf 1990). 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 was 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 were solved to optimality The preponderance of these problems ranged between 10 and 30 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 about 20 times faster than the IBM PS/2 Model 55sx Because, as already pointed out in §4, we could not guarantee feasibility, only 536 of the 640 problems had

Table 9 **Effects of the Number of Resources $\|R\|$ on Solution Times**

R	1	2	3	4	5	6
μ_{CPU}	0.09	1.10	4.29	23.59	138.51	406.15
σ_{CPU}	0.03	3.00	7.44	30.38	375.25	1125.32

Table 11 **Effects of the Number of Jobs J on Solution Times**

J	10	20	30	40
μ_{CPU}	0.06	0.32	23.59	942.09
σ_{CPU}	1.03	0.22	30.38	1439.36

Copyright © 2001 All Rights Reserved

Table 12 **Frequency Distribution of Solution Times for Multi-mode Instances Under Full Factorial Design**

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

a feasible solution. The average time to find and verify the optimal solution was 74.31 seconds. The minimal and maximal time was less than 0.5 seconds and 2016.25 seconds, respectively. Table 12 gives the frequency distribution of the solution times.

The experiment can be summarized as follows: With an increasing resource factor problems become harder. Solution times are far more sensitive to RF_N than to RF_R. When the resource factor of the renewable (nonrenewable) resources is changed from 0.5 to 1.0 the average CPU-time increases from 62.14 (8.10) seconds to 85.70 (124.85) seconds. Moreover, with an increasing resource strength of the nonrenewable resources RS_N the average CPU-time decreases from 363.13 to 3.57 seconds. For the renewable resources the reverse is true; the average CPU-time increases from 54.96 to 95.53 seconds, that is, problems become harder to solve with an increasing availability. If one recalls the results for the single-mode case, this is quite unexpected. A more thorough study of the results provides an explanation. In 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 ($RS_N \leq 0.5$) the effect reverses. Due to the strong effect of RS_N the mean solution times of the different levels of RS_R only show the tendency for scarce nonrenewable resources. A detailed discussion of the computational results is presented in Kolisch, Sprecher and Drexl (1992) and Sprecher (1994).

To sum it up, we could not reproduce the promising results provided by Patterson et al. (cf. 1990) 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 (cf. Sprecher 1994).

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. The method essentially relies on three concepts: A network construction procedure which is based on the definition of a network, the resource factor as a measure of the density of the coefficient matrix as well as the resource strength, which expresses the degree of availability of the resources. All three concepts are very simple and allow to discriminate between easy and hard instances. Moreover, due to their generality they are applicable to other (network) optimization problems and thus are topics of great concern to a wide audience.

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 and, furthermore, that 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.*

* We thank Erik Demeulemeester, Katholieke Universiteit Leuven, for providing us with the code of his algorithm. Furthermore, we wish to thank Sonke Hartmann for his support in implementing the algorithms. Moreover we are indebted to the Associate Editor and three anonymous referees for helpful comments on improving the readability of the paper.

References

Alvarez-Valdes, R. and J. M. Tamarit, "Heuristic Algorithms for Resource-constrained Project Scheduling: A Review and an Empirical Analysis," in R. Slowinski and J. Weglarz (Eds.), *Advances in Project Scheduling*, Elsevier, Amsterdam (1989), 113–134.

Copyright © 2001 All Rights Reserved

Bell, C E and J Han, "A New Heuristic Solution Method in Resource-constrained Project Scheduling," *Naval Res Logistics*, 38 (1991), 315–331

Blazewicz J, W Cellary, R Slowinski and J Weglarz, *Scheduling Under Resource Constraints—Deterministic Models* Blazer, Basel 1986

Cooper, D F, Heuristics for Scheduling Resource-constrained Projects An Experimental Investigation, *Management Sci*, 22 (1976), 1186–1194

Davis, E W 'Project Network Summary Measures Constrained-resource Scheduling, *AIIE Transactions*, 7 (1975), 132–142

Demeulemeester, E and W Herroelen, "A Branch-and-Bound Procedure for the Multiple Resource-constrained Project Scheduling Problem," *Management Sci*, 38 (1992) 1803–1818

——, B Dodin and W Herroelen, "A Random Activity Network Generator," *Oper Res*, 41 (1993), 972–980

Drexl, A, "Scheduling of Project Networks by Job Assignment," *Management Sci*, 37 (1991), 1590–1602

Elmaghraby, S E, *Activity Networks Project Planning and Control by Network Models* Wiley, New York, 1977

—— and W S Herroelen, "On the Measurement of Complexity in Activity Networks," *European J of Operational Res*, 5 (1980), 223–234

Kaimann, R A, "Coefficients of Network Complexity' *Management Sci*, 21 (1974), 172–177

Karp, R M, "Reducibility Among Combinatorial Problems," in R E Miller and J W Thatcher (Eds), *Complexity of Computer Applications*, Plenum Press, New York, 1972, 85–104

Kolisch, R A Sprecher and A Drexl, 'Characterization and Generation of a General Class of Resource constrained Project Scheduling Problems Easy and Hard Instances" Manuskripte aus den Instituten fur Betriebswirtschaftslehre, 301, Kiel 1992

Kurtulus, I S and E W Davis, "Multi-Project Scheduling Categorization of Heuristic Rules Performance," *Management Sci*, 28 (1982), 161–172

Pascoe, T L, 'Allocation of Resources," *C P M Revue Francaise Recherche Operationelle*, 38 (1966), 31–38

Patterson, J H, 'A Comparison of Exact Approaches for Solving the Multiple Constrained Resource, Project Scheduling Problem" *Management Sci*, 30 (1984), 854–867

—— R Slowinski, B Talbot and J Weglarz, "An Algorithm for a General Class of Precedence and Resource Constrained Scheduling Problems,' in R Slowinski and J Weglarz (Eds), *Advances in Project Scheduling*, Elsevier, Amsterdam, (1989), 3 28

——, —— —— and ——, "Computational Experience with a Backtracking Algorithm for Solving a General Class of Precedence and Resource-constrained Scheduling Problems,' *European J of Operational Res*, 49 (1990), 68–79

Pritsker, A A B, W D Watters and P M Wolfe, "Multiproject Scheduling with Limited Resources A Zero-One Programming Approach," *Management Sci*, 16 (1969), 93–108

Sampson, S E and E N Weiss "Local Search Techniques for the Resource Constrained Project Scheduling Problem," *Naval Res Logistics*, 40 (1993), 665–675

Schrage, L, 'A More Portable Fortran Random Number Generator,' *ACM Transactions on Mathematical Software* 5 (1979), 132–138

Sprecher, A, "Resource-constrained Project Scheduling Exact Methods for the Multi-Mode Case," *Lecture Notes in Economics and Mathematical Systems*, 409 (1994), Springer, Berlin

Stinson, J P, E W Davis and B M Khumawala "Multiple Resource-constrained Scheduling Using Branch and Bound," *AIIE Transactions*, 10 (1978) 252–259

Talbot, F B, "Resource-constrained Project Scheduling with Time-resource Tradeoffs The Nonpreemptive Case,' *Management Sci*, 28 (1982), 1197 1210

Accepted by Thomas M Liebling, received December 1992 This paper has been with the authors 7 months for 2 revisions

Copyright © 2001 All Rights Reserved