Drexl, Andreas; Grünewald, Jürgen

Nonpreemptive multi-mode resource-constrained project scheduling

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 236

Provided in Cooperation with:
Christian-Albrechts-University of Kiel, Institute of Business Administration

Nr. 236

Nonpreemptive Multi-Mode Resource-Constrained Project Scheduling

Andreas Drexl *

Jürgen Grünewald **

Oktober 1989

* Andreas Drexl, Institut für Betriebswirtschaftslehre, Christian-Albrechts-Universität zu Kiel, Olshausenstraße 40, 2300 Kiel 1, F.R.G.

** Jürgen Grünewald, Institut für Betriebswirtschaftslehre, Technische Hochschule Darmstadt, Hochschulstraße 1, 6100 Darmstadt, F.R.G.

Abstract: This paper addresses methods for formulating and solving a general class of nonpreemptive resource-constrained project scheduling problems in which job durations are discrete functions of committed renewable, nonrenewable and doubly-constrained resources (multi-mode time resource tradeoff). We present a stochastic scheduling method with which these problems may be solved to suboptimality in an efficient way. Cumputational results demonstrate that this method is highly superior to other well-known existing deterministic scheduling rules. Extensions to problems in which job-specific (demand) resource profiles are varying with time, in addition to time-varying supply resource profiles, are discussed as well.

Traditional multiple resource-constrained project scheduling problems of either the preemptive or the nonpreemptive type have been restricted to the case in which each job has to be performed in exactly one predefined way (single-mode case). Thereby each job may be characterized by a unique duration and a singular collection of resource requirements that have to be met each time period the job is being processed (Bartusch et al. [3], Christofides et al. [6], Davis and Patterson [7], Radermacher [16], Slowinski [17], Stinson et al. [19], Talbot and Patterson [21]).

More recently efforts have been made to formulate and to solve the more general multi-mode preemptive project scheduling problem where job durations are functions of consumed resources (Blazewicz et al. [5], Slowinski [17, 18], Weglarz [22]). Regarding the formulation and solution of nonpreemptive project scheduling problems where job durations are discrete functions of job performance modes in the last decade serious efforts have been documented in literature (Domschke and Drexl [8], Elmaghraby [10], Patterson et al. [14], Talbot [20]).

In this paper we consider the problem of nonpreemptively minimizing project makespan subject to job completion constraints, precedence relations as well as resource restrictions, where the resources are distinguished to be renewable, nonrenewable or doubly-constrained (Slowinski [18], Weglarz [22], Weiss [23]). If a resource is available in limited quantities each time period, this resource is called renewable (resource usage). If total availability of a resource over (part of) the life of the project is constrained, it is called nonrenewable (resource consumption). Finally, resources are defined as doubly - constrained if both their per-period and their total availability are limited (resource usage and consumption).

The remainder of the paper is organized as follows: The next section presents the IP formulation of the scheduling problem. Then, we discuss existing solution techniques and present our stochastic scheduling method. Later on we computationally compare all these solution techniques. Finally we extend the IP formulation to the case where each job has a specific resource demand varying with time and discuss the computational implications. In the last section we offer extensions, conclusions and directions for further research.

Scheduling Model

Without loss of generality we assume a project with one unique source and one unique sink being depicted by an acyclic activity-on-node graph where jobs are numerically labeled such that successor jobs always have higher numbers (labels) than all their predecessors. Specifically, the following assumptions may hold:

- The project consists out of jobs i = 1,...,I. i = 1 (i = I) is the unique source (unique sink). $\mathcal{V}_i$ is the set of all immediate predecessors of job i.
- Job i may be performed in one of the modes j = 1,...,M_i . Each job, once initiated in a specific mode, must be finished without changing mode.
- Scheduling job i in mode j takes d_{ij} time units (duration).
- There are r = 1,...,R renewable resources, where resource r is available with κ^{ρ}_{rt} resource units in period t. Scheduling job i in mode j uses k^{ρ}_{ijr} resource units per period from resource r.
- Nonrenewable resource r = 1,...,N is available with κ^{ν}_{r} resource units during project life-cycle. Scheduling job i in mode j consumes k^{ν}_{ijr} resource units per period from resource r.
- Scheduling must take place regarding r = 1,...,D doubly-constrained resources, where there are κ^{δ}_{rt} resource units available in period t and κ^{δ}_{r} units in total. Processing job i in mode j uses / consumes k^{δ}_{ijr} resource units per period from doubly-constrained resource r.

Assuming an upper bound T for project makespan to be known in advance (heuristically determined), we may calculate earliest and latest start times ES_i and LS_i as well as finish times EF_i and LF_i of job i by traditional forward and backward recursion using shortest durations $dmin_i := \min \{d_{ij} \mid j = 1,...,M_i\}$ for all jobs i. Thus $[EF_i , LF_i]$ represents the maximum time interval within which the project may be finished without violating resource constraints (see below).

Defining variables

$$x_{ijt} := \begin{cases} 1, \text{ job } i \text{ is scheduled in mode } j \text{ and finished in period } t \\ 0, \text{ otherwise} \end{cases}$$

a binary program is formulated as follows using the general framework given in [15]:

$$\min \sum_{j=1}^{M_I} \sum_{t=EF_I}^{LF_I} t \cdot x_{Ijt} \tag{1}$$

Subject to:

$$\sum_{j=1}^{M_i} \sum_{t=EF_i}^{LF_i} x_{ijt} = 1 \quad (i=1,...,I) \tag{2}$$

$$\sum_{j=1}^{M_h} \sum_{t=EF_h}^{LF_h} t \cdot x_{hjt} \leq \sum_{j=1}^{M_i} \sum_{t=EF_i}^{LF_i} (t - d_{ij})\, x_{ijt} \quad (i=1,...,I;\ h \in \mathcal{V}_i) \tag{3}$$

$$\sum_{i=1}^{I} \sum_{j=1}^{M_i} k_{ijr}^{\rho} \sum_{q=t}^{t+d_{ij}-1} x_{ijq} \leq \kappa_{rt}^{\rho} \quad (r=1,...,R;\ t=1,...,LS_I) \tag{4}$$

$$\sum_{i=1}^{I} \sum_{j=1}^{M_i} k_{ijr}^{\nu} d_{ij} \sum_{t=EF_i}^{LF_i} x_{ijt} \leq \kappa_r^{\nu} \quad (r=1,...,N) \tag{5}$$

$$\sum_{i=1}^{I} \sum_{j=1}^{M_i} k_{ijr}^{\delta} \sum_{q=t}^{t+d_{ij}-1} x_{ijq} \leq \kappa_{rt}^{\delta} \quad (r=1,...,D;\ t=1,...,LS_I) \tag{6}$$

$$\sum_{i=1}^{I} \sum_{j=1}^{M_i} k_{ijr}^{\delta} d_{ij} \sum_{t=EF_i}^{LF_i} x_{ijt} \leq \kappa_r^{\delta} \quad (r=1,...,D) \tag{7}$$

$$x_{ijt} \in \{0,1\} \quad (i=1,...,I;\ j=1,...,M_i;\ t=1,...,LF_I) \tag{8}$$

(1) forces the sink job and thus the project being finished as early as possible. (2) represent job completion constraints. (3) states precedence relations between related jobs. (4) and (5) correspond to resource constraints regarding renewable and nonrenewable resources. (6) and (7) secure feasibility with respect to usage and consumption of doubly-constrained resources. It should be noted that (6) and (7) could be incorporated into (4) and (5) appropriately.

Solution Procedures

(1)-(8) may be solved to optimality using the *exact method* of Patterson et al. [14]. This algorithm is an enumerative type of branch and bound method. It simultaneously decides about job-sequencing (which job should preceed others?) and job-mode-assignment (which mode should be assigned to which job ?). Beginning with all jobs being unassigned ($x_{ijt} = 0$ for all i, j, t) the algorithm starts by selecting one job as a candidate for being scheduled as early as possible. The algorithm always builds precedence and resource feasible partial schedules (solutions). "Partial" indicates that not all jobs have currently been scheduled (corresponds to "$\leq$" instead of " =" in (2)). Scheduling jobs is equivalent to augmenting the partial feasible solution. Enumeration is done in a LIFO-implicit way, i.e. partial feasible schedules are augmented as long as neither precedence nor resource infeasibilities occur; in both cases backtracking is done. Due to the unavailability of strong lower bounding procedures for truncating the search tree the computational behaviour of this algorithm is rather poor (see below).

Heuristic scheduling rules adopt a general operating scheme which may be characterized in a simple way as follows (for more details see Kurtulus and Narula [12]): Each job is a member of one of the sets (states): "Finished", "active", "eligible", and "ineligible". A job is termed "finished" when its scheduling has been terminated. When a job is scheduled it becomes "active". A job will be called "eligible" when all its predecessors are scheduled; otherwise, it is considered "inelegible".

There exist a multitude of different ways for constructing *deterministic scheduling rules* by using one or several of the various criteria characterizing model (1)-(8). Following Talbot [20] who presented and compared eight heuristic scheduling rules, the rule MIN LF_i was shown to behave best regarding average quality of solutions:

MIN LF_i The eligible job i with minimum latest finish time LF_i is scheduled first.

The latest finsish times LF_i taken for determining MIN LF_i are calculated by taking

$$T := \sum_{i=1}^{I} \max \{d_{ij} \mid j = 1,...,M_i\}$$

as an upper bound for project makespan and performing a traditional critical path analysis as indicated above using shortest durations $dmin_i$.

Once an eligible job i has been selected according to MIN LF_i for being scheduled next, a quite greedy way of assigning a job mode is to take the mode j with $d_{ij} := \min \{d_{ik} \mid k = 1,...,M_i\}$ and to schedule job i as early as possible. According to (preliminary) comoputaional results not reported here in greater detail this greedy rule performed rather poor especially in cases of scarce renewable resources, i.e. it could not find existing feasible solutions quite oftenly. (It should be emphasized that this poor behaviour is in principal intrinsic to all heuristic rules which try to construct schedules deterministically.) In order to overcome this disadvantage while keeping the deterministic behaviour we modified / augmented MIN LF_i as follows: Modes $j = 1,...,M_i$ are selected in increasing order for being assigned to job i until a mode is found which does not violate resource restrictions. If no such mode can be found the deterministic rule is unable to determine a feasible solution for the problem.

Now we are going to present a *stochastic scheduling method* which overcomes the deficiencies associated with deterministic methods up to a large extent. (The method has been successfully used in Drexl [9] to solve assignment type project scheduling problems.) The stochastic nature of these method emerges from using some criteria measuring the impacts of job selection and mode assignment in a probabilistic way.

More precisely we calculate

$$\omega_{ij} := (\max \{d_{ik} \mid k = 1,...,M_i\} - d_{ij} + \epsilon)^{\alpha} \qquad (9)$$

for $i \in EJ$ where EJ corresponds to the set of eligible jobs and job modes $j = 1,...,M_i$ are taken appropriately. (9) measures the worst-case consequence of assigning mode j *not* to job i with respect to job durations. $\epsilon > 0$ secures ω_{ij} to be positive and $\alpha \geq 0$ transformes the term (.) in an exponential way thus diminishing or enforcing the differences between the mode dependent job durations for $\alpha < 1$ or $\alpha > 1$, respectively. It suggests itself to use ω_{ij} for stochastic job selection and (or) mode assignment with probabilities proportional to ω_{ij} for all $i \in EJ$ and $j = 1,...,M_i$.

Alternatively we may use

$$\gamma_i := (\max \{LF_k \mid k \in EJ\} - LF_i + \epsilon)^{\alpha} \qquad (10)$$

for selecting $i \in EJ$ randomly with probabilities proportional to γ_i for all $i \in EJ$ in a first stage and then assign mode j to job i at random based on ω_{ij} in a second stage.

For convenience we will denote both stochastic job selection and mode assignment procedures with JOSEMODA, whereas JOSEMODA1 and JOSEMODA2 specifically denote, whenever necessary, the one based on (9) and (10), respectively.

Formally the stochastic scheduling method may be described as follows:

1. Set EJ := {1} and determine LF_i (i = 1,...,I) by modified critical path analysis.
2. Select job $i \in EJ$ and assign mode j to job i randomly using procedure JOSEMODA; schedule job i as early as possible regarding precedence relations only.
3. Update EJ and LF_i appropriately; if EJ = ϕ, then STOP else goto 2.
4. Check feasibility of the generated schedule with respect to resource restrictions (4)-(7) and STOP.

The procedure does not check resource feasibility of partial schedules due to the following reasons: Scaling arrays representing renewable, nonrenewable and doubly-constrained resource availabilities dynamically whenever a specific resource is used and / or consumed takes substantial CPU-time. Preliminary computational results not reported here in detail showed that updating left-over capacities in a dynamic fashion needs three to four of CPU-time compared with the described version which checks resource feasibility only in the final step when all jobs have been scheduled.

Disregarding the result of one pass through the above procedure (i.e. a feasible or an infeasible schedule / solution) this algorithm should be applied to a specific data set several times in order to (hopefully) generate a lot of feasible schedules und thus a near optimal one, too.

Computational Results

The algorithms have been coded in TURBO PASCAL and implemented on a Personal Computer (XT compatible). The test data generated for comparative purposes may be characterized as follows (project summary measures):

- The problem size, in terms of the number of jobs, is the first project summary measure.
- The network complexity C = number of arcs (precedence relations) / number of nodes (jobs) affects the performance of scheduling procedures and thus must be part of the experimental design.
- The number of modes of job i is generated randomly for all data sets according to $2 \leq M_i \leq 4$ with 3 as average number of job modes.
- The total number of resource types R (≤ 5), N (≤ 3), and D (≤ 3) is fixed (within each "problem class") in advance.
- The (integer) duration d_{ij} of job i being scheduled in mode j is generated at random according to $5 \leq d_{ij} \leq 10$.
- Job-mode dependent (integer) resource demands (requirements) are randomly generated such that $0 \leq k^{\rho}_{ijr}, k^{\nu}_{ijr}, k^{\delta}_{ijr} \leq 5$ for all resources.
- The renewable resource availabilities κ^{ρ}_{rt} are determined by multiplying the peak resource requirement

$$\kappa^{peak}_{r} := \max_{i \in I} \max_{j \in J_i} \max \{k^{\rho}_{ijr} \mid \forall\, t\}$$

with K1 thus getting

$$\kappa^{\rho}_{rt} := \kappa^{peak}_{r} \cdot K1$$

with $1.5 \leq K1 \leq 3.3$ (within this section capacities are constant over time for each resource ; the time-varying case see below). κ^{δ}_{rt} is treated analogously.

- The nonrenewable resource capacities κ^{ν}_{r} are calculated by multiplying $\sum_{i=1}^{I} k^{\nu}_{i1r}$ (mode one) with K2 ($0.7 \leq K2 \leq 1.1$) for each resource r. κ^{δ}_{r} is treated analogously.

In order to evaluate the CPU-times required by the above exact method, we generated and solved to optimality 100 data sets with the following project summary measures:

I = 10, C = 1.5, R = 3, N = 1, D = 0, K1 = 2.0, and K2 = 0.9 (our implementation of the exact method does not handle doubly-constrained resources explicitely; furthermore it is only able to deal with one nonrenewable resource ($N \leq 1$)).

Table 1 presents average times (input / output excluded). It can be seen, that CPU-times are (within the same problem class) highly dependent on the structure of the data sets. In 22% of the test problems it took only upt to 2 sec to find and to verify the optimum solution; but in 21% it took more than 8 min (with 56 min as largest CPU-time) to determine the optimum solution.

Table 1: Distribution of CPU-times for the exact method

CPU-time	percentage
0-2 sec	22%
>2-20 sec	21%
>20-60 sec	17%
>1-3 min	10%
>3-8 min	9%
>8-20 min	12%
>20-60 min	9%

In contrast to this the CPU-times required by the rule MIN LF_i and by JOSEMODA1 are rather low. Table 2 presents minimum, mean, and maximum CPU-times (input / output times excluded) in sec (MIN, MEAN, and MAX) necessary to heuristically solve 10 data sets within each problem class (characterized by the number of jobs I) where $\epsilon = \alpha = 1$ and the other parameters are the same as in table 1. JOSEMODA1 has been executed 10 times for each data set. (It should be noted that the above choice of ϵ which seems to be rather arbitrary is uncrucial for the behavior of the algorithm. The effect of a varying α will be shown below in tables 5 to 7.)

Table 3 compares the quality of solutions obtained with JOSEMODA1 and JOSEMODA2 (for projects with I = 10, C = 1.5, R = 3, N = 1, D = 0). For each combination of K1 and K2 (problem class) 10 projects have been generated and solved iteratively 10 times (using $\epsilon = \alpha = 1.0$). "Better", "equal", and "worse" denote the percentage of cases in which JOSEMODA1 produced better, equal, or worse results than JOSEMODA2, respectively.

Table 2: CPU-times of MIN LF_i and JOSEMODA1

problem class (I)	MIN LF_i MIN	MEAN	MAX	JOSEMODA1 MIN	MEAN	MAX
10	0.82	0.98	2.03	0.44	0.85	1.43
20	1.78	1.93	2.13	1.70	2.83	4.67
30	1.81	3.79	5.17	2.91	4.86	5.50
40	5.02	5.33	8.73	5.54	7.54	10.99
50	3.35	6.54	15.33	9.98	13.20	23.42
60	2.37	8.91	12.85	12.25	17.43	31.91
70	5.53	11.37	14.77	14.49	22.91	37.20
80	4.32	16.62	23.43	16.77	26.41	39.09

Table 3: Comparison 1 of JOSEMODA1 and JOSEMODA2

problem class	better	equal	worse	Σ
K1=2.5/K2=1.1	75%	15%	10%	100%
K1=2.0/K2=0.9	81%	13%	6%	100%
K1=1.5/K2=0.7	89%	5%	6%	100%

Table 4 presents average percentage deviations of objective function values (from the optimum) obtained by applying JOSEMODA1 and JOSEMODA2 10 times to 10 projects with I = 10, C = 1.5, R = 3, N = 1, D = 0, K1 = 2.0, K2 = 0.9 (once more for $\epsilon = \alpha = 1.0$).

Table 4: Comparison 2 of JOSEMODA1 and JOSEMODA2

procedure	percentage deviation from optimum 0%	>0-2%	>2%	Σ
JOSEMODA1	58%	28%	14%	100%
JOSEMODA2	54%	19%	27%	100%

Tables 3 and 4 show that JOSEMODA1 is superior to JOSEMODA2; thus only the first one will be investigated in more detail now.

Tables 5 to 7 provide results which demonstrate the influence of a varying α on the behavior of JOSEMODA1. In each case (row of tables 5 to 7) JOSEMODA1 has been iteratively executed until 100 feasible solutions could be found. The problem that has been treated in all cases is characterized by I = 10, C = 1.5, R = 3, N = 1 and D = 0. Each entry of the tables represents the number (percentage) of feasible solutions, whose objective function value had an appropriate percentage deviation from optimum.

Table 5: Variation of α (K1 = 2.5 / K2 = 1.1)

α	percentage deviation from optimum							#trials
	0-2%	>2-4%	>4-6%	>6-8%	>8-10%	>10-12%	>12%	
0.0	1	6	20	21	35	14	3	202
0.5	6	24	28	24	15	3	-	224
1.0	27	43	24	2	4	-	-	234
1.5	58	22	17	-	3	-	-	299
2.0	84	15	1	-	-	-	-	361

Table 6: Variation of α (K1 = 2.0 / K2 = 0.9)

α	percentage deviation from optimum							#trials
	0-2%	>2-4%	>4-6%	>6-8%	>8-10%	>10-12%	>12%	
0.0	15	5	16	27	20	16	1	251
0.5	14	13	22	27	11	8	5	279
1.0	12	16	37	13	16	4	2	273
1.5	52	26	19	2	1	-	-	295
2.0	77	11	9	1	1	1	-	296

Table 7: Variation of α (K1 = 1.5 / K2 = 0.7)

α	percentage deviation from optimum							#trials
	0-2%	>2-4%	>4-6%	>6-8%	>8-10%	>10-12%	>12%	
0.0	-	1	26	33	29	9	2	671
0.5	8	11	39	26	13	-	3	789
1.0	36	36	21	6	4	1	-	2156
1.5	46	23	25	5	1	-	-	2698
2.0	55	30	12	3	-	-	-	3779

Tables 5, 6, and 7 present results for K1 = 2.5 / K2 = 1.1, K1 = 2.0 / K2 = 0.9, K1 = 1.5 / K2 = 0.7, respectively. The last column of each of the three tables gives the number of trials (executions of JOSEMODA1 with ϵ = 1.0) which were necessary in order to find 100 feasible solutions. The results may be interpreted as follows:

- With increasing α more trials are necessary in order to generate a prespecified number of feasible solutions.
- The probability of finding a feasible solution is decreasing with increasing α, whereas the quality of solutions is increasing as well.
- The percentage deviations of the objective function value from the optimum one are decreasing with increasing α.
- With decreasing resource availabilities more difficulties arise to find feasible solutions thus needing more trials.

Summarizing these observations a general advice for an appropriate choice of α may be given as follows: For a given data set one does not know in advance the scarcity of resources. Thus one should start with a rather high α (say α = 2.0), perform some hundreds of trials (depending on the amount of available computer resources), reduce α as indicated above and so on.

Table 8 provides results of a comparison of MIN LF_i with JOSEMODA1 for projects with I = 10, C = 1.5, R = 3, N = 3, and D = 2 as well as varying resource availabilities. For each combination of K1 and K2 five data sets have been generated and solved with both procedures, where the number of trials of JOSEMODA1 was set equal to 30 and α = ϵ = 1.0. Each entry #1 / #2 corresponds to the number of data sets, for which MIN LF_i (#1) and JOSEMODA2 (#2) were unable to find a feasible solution, respectively.

These results demonstrate that the stochastic procedure JOSEMODA1 has a much greater capability of finding feasible solutions than the deterministic procedure MIN LF_i , even in the case of rather scarce renewable and nonrenewable resources.

Table 8: Comparison 1 of MIN LF_i and JOSEMODA1

K2 / K1	1.0	0.9	0.8	0.7
3.3	1/0	4/0	4/0	4/2
3.0	4/0	4/2	5/1	5/2
2.7	3/0	5/0	4/2	5/1
2.4	5/0	4/0	5/4	4/3
2.1	4/0	3/0	5/2	5/3
1.8	3/0	5/2	4/3	5/4
1.5	3/0	5/0	5/4	5/5

Table 9 compares the quality of solutions for those cases of table 8, in which MIN LF_i could find a feasible solution (23 from $4 \cdot 7 \cdot 5 = 140$ in total). Percentage deviations of objective function values (OFV) are mesured according to

$$[(OFV_{MIN\ LF_i} - OFV_{JOSEMODA1}) / OFV_{JOSEMODA1}],$$

where $OFV_{JOSEMODA1}$ corresponds to the best objective function value found within 30 trials. These results show that JOSEMODA1 produces results which are, except for rather few cases, much more better than those produced by MIN LF_i .

Table 9: Comparison 2 of MIN LF_i and JOSEMODA1

percentage deviation	-2>0%	0%	>0-4%	>4-8%	>8-12%	>12%
quantity	2	1	6	8	4	2

Tables 8 and 9 demonstrate that JOSEMODA1 is superior to MIN LF_i with respect to two important capabilities of heuristics, i.e. to find existing feasible solutions and to produce "good" solutions rather quickly. These results are rather typical with respect to the behavior of both methods. They remain unchanged especially for projects with a larger job number I and with other network complexities C.

Resource Requirements Varying with Time

Model (1)-(8) covers project scheduling problems with time-varying supply resource profiles. Now we are going to model the situation where the job resource requirements are time-varying, too.

Specifically $k^{\rho}_{ijr\tau}$, $k^{\nu}_{ijr\tau}$ and $k^{\delta}_{ijr\tau}$ ($\tau = 1,...,d_{ij}$) denotes the requirements of renewable, nonrenewable and doubly-constrained resources which are necessary to schedule job i in mode j in period $t+\tau$, where t corresponds to the period (unknown in advance) in which scheduling of job i will start. Regarding (1)-(8) we thus easily get the generalized model (11)-(18):

$$\min \sum_{j=1}^{M_I} \sum_{t=EF_I}^{LF_I} t \cdot x_{Ijt} \tag{11}$$

Subject to:

$$\sum_{j=1}^{M_i} \sum_{t=EF_i}^{LF_i} x_{ijt} = 1 \quad (i=1,...,I) \tag{12}$$

$$\sum_{j=1}^{M_h} \sum_{t=EF_h}^{LF_h} t \cdot x_{hjt} \le \sum_{j=1}^{M_i} \sum_{t=EF_i}^{LF_i} (t - d_{ij})\, x_{ijt} \quad (i=1,...,I;\ h \in \mathcal{V}_i) \tag{13}$$

$$\sum_{i=1}^{I} \sum_{j=1}^{M_i} \sum_{q=t}^{t+d_{ij}-1} k^{\rho}_{ijr(q-t+1)}\, x_{ijq} \le \kappa^{\rho}_{rt} \quad (r=1,...,R;\ t=1,...,LS_I) \tag{14}$$

$$\sum_{i=1}^{I} \sum_{j=1}^{M_i} \sum_{\tau=1}^{d_{ij}} k^{\nu}_{ijr\tau} \sum_{t=EF_i}^{LF_i} x_{ijt} \le \kappa^{\nu}_{r} \quad (r=1,...,N) \tag{15}$$

$$\sum_{i=1}^{I} \sum_{j=1}^{M_i} \sum_{q=t}^{t+d_{ij}-1} k^{\delta}_{ijr(q-t+1)} \, x_{ijq} \leq \kappa^{\delta}_{rt} \quad (r=1,...,D;\ t=1,...,LS_I) \tag{16}$$

$$\sum_{i=1}^{I} \sum_{j=1}^{M_i} \sum_{\tau=1}^{d_{ij}} k^{\delta}_{ijr\tau} \sum_{t=EF_i}^{LF_i} x_{ijt} \leq \kappa^{\delta}_{r} \quad (r=1,...,D) \tag{17}$$

$$x_{ijt} \in \{0,1\} \quad (i=1,...,I;\ j=1,...,M_i\ ;\ t=1,...,LF_I) \tag{18}$$

It is simple to see that MIN LF_i and JOSEMODA1 can easily be generalized in such a way that model (11)-(18) can be solved. In contrast to this it would be - though possible in principal - rather involved (and inefficient as well) to generalize the exact method accordingly. Thus we compare MIN LF_i and JOSEMODA1 only.

The experimental design described above is generalized as follows: At first job-specific resource requirements are generated at level λ ($0 \leq \lambda \leq 3$; constant over time). Then each resource level λ is changed in a random fashion such that ($\max\{0, \lambda-2\} \leq k^{\rho}_{ijr\tau}$, $k^{\nu}_{ijr\tau}$, $k^{\delta}_{ijr\tau} \leq \lambda + 2$) holds for all $\tau = 1,...,d_{ij}$. Once resource requirements are known resource availabilities κ^{ρ}_{rt}, κ^{ν}_{r}, κ^{δ}_{rt}, and κ^{δ}_{r} are determined appropriately.

MIN LF_i and JOSEMODA1 have been compared analogously to tables 8 and 9. The results are rather similar. They show (once again) that JOSEMODA1 has a much greater capability of finding "good" feasible solutions than MIN LF_i.

Computational times of both procedures are nearly identical with those presented in table 2 because there is only little additional effort necessary for updating arrays in the case of time-varying resource requirements.

Summary and Conclusions

The paper presents models for formulating nonpreemptive project scheduling problems with general resource availabilities and requirements as well as multi-mode time resource tradeoffs. For the solution of this model, a stochastic scheduling method is presented which outperformes traditional deterministic scheduling rules drastically.

The method is general and flexible enough for being applied to project planning problems which incorporate some additional features such as multiple projects (Kurtulus and Davis [11], Mohanty and Siddiq [13]), time windows (Bartusch et al. [3]), and distributed simulation concepts (Arora and Sachdewa [2]). Thus it is supposed to be well suited for DSS approaches to project scheduling (Anthonisse et al. [1]), Bartusch et al. [4]) and future work should show that it may be usefully embedded in DSS.

References

[1] Anthonisse, J.M., Van Hee, K.M., Lenstra, J.K., "Resource-Constrained Project Scheduling: an International Exercise in DSS Development", *Decision Support Systems*, Vol. 4 (1988), pp. 249-257.

[2] Arora, R.K., Sachdewa, R.K., "Distributed Simulation of Resource Constrained Project Scheduling", *Comput. & Ops. Res.*, Vol. 16 (1989), pp. 295-304.

[3] Bartusch, M., Möhring, R.H. Radermacher, F.J., "Scheduling Project Networks with Resource Constraints and Time Windows", *Annals of Operations Research*, Vol. 16 (1988), pp. 201-240.

[4] Bartusch, M., Möhring, R.H., Radermacher, F.J., "Design Aspects of an Advanced Model- Oriented DSS for Scheduling Problems in Civil Engineering", to appear in *Decision Support Systems.*

[5] Blacewicz, J., Cellary, W., Slowinski, R., Weglarz, J., "*Scheduling Under Resource Constraints - Deterministic Models*", Basel 1986 (Annals of Operations Research, Vol. 7).

[6] Christofides, N., Alvarez-Valdes, R., Tamarit, J.M., "Project Scheduling with Resource Constraints: A Branch and Bound Approach", *European Journal of Operational Research*, Vol. 29 (1987), pp. 262-273.

[7] Davis, E.W., Patterson, J.H., "A Comparison of Heuristic and Optimum Solutions in Resource-Constrained Project Scheduling", *Management Science*, Vol. 21 (1975), pp. 944-955.

[8] Domschke, W., Drexl, A., "Deterministische Modelle zur Planung und Kontrolle von Projekten mit Kapazitätsrestriktionen", Discussion Paper, Technische Hochschule Darmstadt, February 1989.

[9] Drexl, A., "Scheduling of Project Networks by Job Assignment", Discussion Paper, Technische Hochschule Darmstadt, February 1989.

[10] Elmaghraby, S.E., "*Activity Networks: Project Planning and Control by Network Models*", New York 1977.

[11] Kurtulus, I.S., Davis, E.W., "Multi-Project Scheduling: Categorization of Heuristic Rules Performance", *Management Science*, Vol. 28 (1982), pp. 161-172.

[12] Kurtulus, I.S., Narula, S.C., "Multi-Project Scheduling: Analysis of Project Performance", *IIE Transactions*, Vol. 17 (1985), pp. 58-66.

[13] Mohanty, R.P., Siddiq, M.K., "Multiple Projects - Multiple Resources - Constrained Scheduling: Some Studies", *International Journal of Production Research*, Vol. 27 (1989), pp. 261-280.

[14] Patterson, J., Slowinski, R., Talbot, B., Weglarz, J., "An Algorithm for a General Class of Precedence and Resource Constrained Scheduling Problems", in: Slowinski, R., Weglarz, J. (Eds.): *"Advances in Project Scheduling"*, Amsterdam 1989, pp. 3-28.

[15] Pritsker, A.A.B., Watters, W.D., Wolfe, P.M., "Multiproject Scheduling With Limited Resources: A Zero-One Programming Approach", *Management Science*, Vol. 16 (1969), pp. 93-108.

[16] Radermacher, F.J., "Scheduling of Project Networks", *Annals of Operations Research*, Vol. 4 (1985/6), pp. 227-252.

[17] Slowinski, R., "Two Approaches to Problems of Resource Allocation Among Project Activities - A Comparative Study", *Journal of the Operational Research Society*, Vol. 31 (1980), pp. 711-723.

[18] Slowinski, R., "Multiobjective Network Scheduling With Efficient Use of Renewable and Nonrenewable Resources", *European Journal of Operational Research*, Vol. 7 (1981), pp. 265-273.

[19] Stinson, J.P., Davis, E.W., Khumawala, B.M., "Multiple Resource-Constrained Scheduling Using Branch and Bound", *AIIE Transactions*, Vol. 10 (1978), pp. 252-259.

[20] Talbot, F.B., "Resource - Constrained Project Scheduling With Time-Resource Tradeoffs: The Nonpreemptive Case", *Management Science*, Vol. 28 (1982), pp. 1197-1210.

[21] Talbot, F.B., Patterson, J.H., "An Efficient Integer Programming Algorithm With Network Cuts for Solving Resource-Constrained Scheduling Problems", *Management Science*, Vol. 24 (1978), pp. 1163-1174.

[22] Weglarz, J., "Project Scheduling With Continuously Divisible, Doubly - Constrained Resources", *Management Science*, Vol. 27 (1981), pp. 1040-1053.

[23] Weiss, E.N., "An Optimization Based Heuristic for Scheduling Parallel Project Networks with Constrained Renewable Resources", *IIE Transactions*, Vol. 20 (1988), pp. 137-143.