ECONSTOR
Make Your Publications Visible.

A Service of
ZBW
Leibniz-Informationszentrum Wirtschaft
Leibniz Information Centre for Economics

Jordan, Carsten; Drexl, Andreas

Working Paper — Digitized Version

A comparison of logic and mixed-integer programming solvers for batch sequencing with sequence-dependent setups

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 322

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

Suggested Citation: Jordan, Carsten; Drexl, Andreas (1993) : A comparison of logic and mixed-integer programming solvers for batch sequencing with sequence-dependent setups, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 322, Universität Kiel, Institut für Betriebswirtschaftslehre, Kiel

This Version is available at:
https://hdl.handle.net/10419/155402

Standard-Nutzungsbedingungen:

Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden.

Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.

Terms of use:

Documents in EconStor may be saved and copied for your personal and scholarly purposes.

You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.

If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.

WWW.ECONSTOR.EU

Mitglied der Leibniz-Gemeinschaft

Nr. 322

A Comparison of Logic and Mixed-Integer Programming Solvers for Batch Sequencing with Sequence-Dependent Setups

Jordan, C. and A. Drexl

July 1993

Carsten Jordan, Andreas Drexl, Institut für Betriebswirtschaftslehre, Christian-Albrechts-Universität zu Kiel, Olshausenstr. 40, 24118 Kiel, Germany.

Abstract: A batch sequencing model with sequence-dependent setup-times and -costs is used to compare modelling and solving with two different general solvers. "Conceptual models" are implemented in the constraint propagation / logic programming language CHARME and solved with the PROLOG inference engine. The mixed-integer-programming (MIP) formulation of the same problem is solved with OSL, a state-of-the-art MIP solver. Modelling is easier in CHARME and computational results show, that the first approach outperforms the second one for instances with high capacity utilization.

Keywords: SINGLE MACHINE BATCHING/ SEQUENCING, DEADLINES, SEQUENCE-DEPENDENT SETUP-TIMES AND -COSTS, LOGIC/ MIXED-INTEGER PROGRAMMING.

Often problem solving must be done with general solvers as there is not enough time (or it is not worthwhile) to develop special purpose algorithms. In these cases the user is interested in an easily manageable tool to implement and solve his specific problem. We compare two different general solvers for a certain example problem: a mixed-integer programming (MIP) package and a constraint propagation / logic programming language. Both have the advantage, that no algorithm must be specified, only the problem has to be represented in an appropriate form. We show how the same problem can be modelled differently for both solvers, and we are interested in their ability to encode constraints easily and in their performance.

Batch sequencing problem

We consider a batch sequencing problem where a number of jobs has to be processed on a single machine/facility, each job is available at time zero and must be completed before its deadline. The set of jobs is partitioned into classes, and a sequence-dependent setup-time is needed when switching from jobs in one class to another. Jobs in one class have to be processed according to their earliest deadline ordering. This is motivated by considering jobs of one class as demands of a certain item. We consider four different objectives: First to minimize the sum of sequence dependent setup-costs (between classes) and earliness penalties for each job completed before its deadline (Case I); second, only the sequence-dependent setup-costs (Case II); third only the sum of earliness costs has to be minimized (Case III); fourth - without any objective function - the problem may be viewed as a Constraint Satisfaction Problem (Case IV).

Mixed-integer programming formulations versus logic programming

A problem stated as a MIP-formulation can be solved with an LP-based branch-and-bound algorithm. As nonlinear expressions are not allowed, MIP-formulations often need a lot of decision variables which increases the computational complexity of the underlying branch-and-bound algorithm. The solution times heavily depend on the quality of the LP-bounds which depend on the structure of the model. In general, MIP formulations for scheduling problems are largely intractable and not easy to state. For our tests we use OSL (cf. IBM [9]).

Constraint propagation languages allow a declarative, even a nonlinear problem description. Variables in those languages have an associated domain and constraints are used to reduce the size of the domains and to eliminate infeasible values (constraint propagation). Originally, constraint propagation languages have been developed for constraint satisfaction problems (CSP), where values for each variable (out of its appropriate domain) must be found so that all constraints are satisfied. Normally, constraint propagation itself is not sufficient to find a solution so that enumeration is

necessary, constraints now serve to cut the enumeration tree. Minimization of an objective can be done by finding all solutions through enumeration and keeping the best objective value as an upper bound. So we expect constraint propagation languages to perform well for problems with a small solution space when they are used for optimization. We use the constraint propagation language CHARME (cf. BULL [4]), other logic programming languages are described in e.g. Van Hentenryck [8].

Review of related work

Modelling capabilities and performance of integer programming versus expert systems have been compared for one instance of a course scheduling problem by Dhar/Ranganathan [6]. A survey of MIP formulations for scheduling problems is given in Blazewicz et al. [2], Case III equals a MIP formulation given by Coleman [5] without tardiness. A similar MIP formulation of Cases I and II has been developed by Sielken [12]. The complexity of the batch sequencing problem has been considered by Monma/Potts [10], and by Bruno/Downey [3], who show the feasibility problem to be NP-hard, even for sequence-independent setup-times. Woodruff/Spearman [14] give a similar and extended "conceptual formulation" of the above problem and solve it with a tabu search heuristic. Unal/Kiran [13] address the feasibility problem in the context of order acceptance.

In Section 1.1 we formulate a "conceptual model" of the batch sequencing problem, which is implemented in the constraint propagation / logic programming language CHARME. In Section 1.2 the different MIP formulations are presented. A numerical example illustrates the model in Section 2, and in Section 3 some computational experience is reported. Conclusions follow in Section 4.

1. Model formulations

In the following we present the parameters used for all formulations.

Parameters

J	number of jobs
j	job index $0..J+1$, 0 is the first (dummy) job, $J+1$ the last (dummy) job
N	number of jobclasses (=items)
m	index of the jobclass(=item), $m=1..N$
M_m	set of jobs belonging to jobclass(=item) m
h_m	holding costs per unit time of one unit of jobs in jobclass m
d_j	deadline of job j
p_j	processing time of job j
$e_j = p_j\, h_m$	earliness costs per unit time of job j, $j \in M_m$
st_{mn}	setup-time between jobclasses m and n, $m,n = 1..N$
sc_{mn}	setup-cost between jobclasses m and n, $m,n = 1..N$
ST_{ij}	setup-time between job i and job j, $i = 0..J$, $j = 1..J$
	$ST_{ij} = \begin{cases} 0 & \text{for } i, j \in M_m \\ st_{mn} & \text{for } i \in M_m, j \in M_n,\ m \neq n \end{cases}$
SC_{ij}	setup-cost between job i and job j, $i = 0..J$, $j = 1..J$

$$SC_{ij} = \begin{cases} 0 & \text{for } i, j \in M_m \\ sc_{mn} & \text{for } i \in M_m, j \in M_n, m \neq n \end{cases}$$

B big number

Jobs are labelled from 1 to J, and in each jobclass they are labelled in the earliest deadline order, so that $\forall\ i,j \in M_m$ and $i<j$ we have $d_i<d_j$ (cf. Fig.1 in Section 2). The disjoint subsets M_m define a partition of the set of jobs. Earliness costs represent holding costs, so for each item they are proportional to the processing time and interchanging two jobs of the same item does not alter the earliness costs. The matrices ST_{ij} and SC_{ij} are enlarged matrices of the setup-time and -cost matrices between jobclasses.

1.1 Conceptual model formulation

After defining the following decision variables, a "conceptual model formulation" is given. Brackets [] are sometimes used to better distinguish variables and indexes.

Decision variables

$X_{[j]}$	completion time of job j
S	sequence of jobs,
S_k	job at position k
R	positions (ranks) of jobs,
R_j	position of job j

$$Min \quad \sum_{j=1}^{J} e_{[j]}(d_{[j]} - X_{[j]}) + SC_{[S_{j-1},S_j]} \tag{1}$$

s.t.

$$X_{[j]} \leq d_{[j]} \qquad (j=1..J) \tag{2}$$

$$X_{[S_{k-1}]} + ST_{[S_{k-1},S_k]} + P_{[S_k]} \leq X_{[S_k]} \qquad (k=1..J) \tag{3}$$

$$R_{[i]} < R_{[j]} \qquad (i<j;\ i,j \in M_m;\ m=1..N) \tag{4}$$

$$S_{[R_j]} = j \qquad (j=1..J) \tag{5}$$

$$X_{[0]} = 0,\ S_0 = 0 \tag{6}$$

This formulation can be implemented directly in CHARME. The objective function (1) minimizes the sum of early completion and setup-costs for Case I. For Case II only the setup-costs (earliness costs = 0), for Case III only the earliness cost are minimized (setup-costs=0). Case IV determines a sequence

and completion times feasible w.r.t. (2) - (6). Constraints (2) are the "demand constraints" (each job must be completed before its deadline) and (3) accounts for a correct sequence on the machine: The completion time of the job at position k must be greater than the completion time of the previous job k-1 plus the setup- and its processing time. Constraints (2) and (3) can be combined, i.e.:

$$X_{[S_{k-1}]} \leq \min\left\{d_{[S_{k-1}]}, X_{[S_k]} - ST_{[S_{k-1},S_k]} - P_{[S_k]}\right\}, \quad k = 1..J$$

In (5) the arrays S (sequence) and R (position) are related, and (4) expresses the ordering of jobs in one jobclass in the earliest deadline ordering. With (6) the sequence is initialized. Note that the model can be concisely formulated by using decision variables as *indexes* (cf. Woodruff/Spearman [14]) and is therefore called a conceptual model. This is impossible in MIP-formulations where the variables and their coefficients must be identified in the formulation.

1.2 Mixed-integer programming formulations

The MIP formulations of the above models are less compact. In the equations we have to consider explicitly that each job may be scheduled at (nearly) each position of the sequence, which leads to constraints for each *pair (i,j)*. To achieve the best performance we employ different MIP formulations for different objectives (cases). If sequence-dependent setup-costs have to be considered the binary decision variables must indicate that job j follows *immediately* job i. For Cases III and IV the number of binary variables can be reduced defining them as job j is scheduled (not necessarily immediately) after job i.

MIP formulation Cases I and II (MIP A)

Decision variables

$y_{ij} = \begin{cases} 1 & \text{if job } i \text{ is scheduled } \textit{directly} \text{ before } j \\ 0 & \text{otherwise} \end{cases}$

X_j completion time of job j

$$Min \sum_{j=1}^{J} e_j(d_j - X_j) + \sum_{i=0}^{J} \sum_{\substack{j=1 \\ j \neq i}}^{J} SC_{ij} y_{ij} \tag{7}$$

s.t.

$$\sum_{\substack{j=1 \\ j \neq i}}^{J+1} y_{ij} = 1 \qquad (i=0..J) \tag{8}$$

$$\sum_{\substack{i=0 \\ i \neq j}}^{J} y_{ij} = 1 \qquad (j=1..J+1) \tag{9}$$

$$X_i + ST_{ij} + p_j \leq X_j + B(1 - y_{ij}) \qquad (i=0..J;\ j=1..J+1; j \neq i) \tag{10}$$

$$X_j \le d_j \qquad (j=1..J+1) \qquad (11)$$

$$X_j \ge 0 \qquad (j=1..J+1) \qquad (12)$$

$$y_{ij} \in \{0,1\} \qquad (i=0..J;\ j=1..J+1; j \ne i) \qquad (13)$$

$$y_{ij} = 0 \qquad (i>j;\ i,j \in M_m;\ m=1..N) \qquad (14)$$

$$X_0 = 0 \qquad (15)$$

For Case I both terms of the objective are important, in Case II the earliness costs are omitted (earliness costs = 0). The y_{ij} must define a permutation with 0 as the first, $J+1$ as the last job, so that (8) and (9) are assignment constraints (cf. Blazewicz et al.[2]). For y_{ij} = 1 constraints (10) enforce the sequence on the machine (job i is scheduled *directly* before j). Constraints (11) correspond to (2). Due to the ordering in jobclasses some of the sequencing variables y_{ij} are fixed in advance in (14).

MIP formulation Cases III and VI (MIP B)

Now, sequence dependent setup-costs need not to be considered in the objective. With the decision variables defined as below only half of the binary variables is needed. For Cases III and IV the setup-time matrice is assumed to be triangular, that is $ST_{ij}+ST_{jk} \ge ST_{ik}$. In practice it is reasonable to assume that not two setups can be performed to accomplish one.

Decision variables

$$y_{ij} = \begin{cases} 1 & \text{if job } i \text{ is scheduled before job } j \\ 0 & \text{otherwise } (\Rightarrow \text{job } i \text{ is scheduled } \textit{after} \text{ job } j) \end{cases}$$

---

and the variables X_j defined as above.

$$Min \sum_{j=1}^{J} e_j(d_j - X_j) \qquad (16)$$

s. t.

$$ST_{0j} + p_j \le X_j \le d_j \qquad (j=1..J) \qquad (17)$$

$$X_j - X_i + B(1-y_{ij}) \ge p_j + ST_{ij} \qquad (i=1..J;\ j=i+1..J) \qquad (18)$$

$$X_i - X_j + By_{ij} \ge p_i + ST_{ji} \qquad (i=1..J;\ j=i+1..J) \qquad (19)$$

$$X_j \ge 0 \qquad (j=1..J) \qquad (20)$$

$$y_{ij} \in \{0,1\} \qquad (i=1..J;\ j=i+1..J) \qquad (21)$$

$$y_{ij} = 0 \qquad (i>j;\ i,j \in M_m;\ m=1..N) \qquad (22)$$

In Case III the objective minimizes the earliness costs (setup-costs=0), in Case IV values for X_j and y_{ij} subject to (17) - (22) must be found. Constraints (17) define the time window for the completion time of job j. The constraints (18) and (19) sequence the jobs; originally they have been proposed by Baker [1] and later on extended by Coleman [5] for the case of sequence-dependent setup-times. As in MIP A some of the y_{ij} can be fixed beforehand (22). Omitting the setup-costs in MIP A would give another formulation of Case III, but the performance of MIP B is much better. Reversely, with a lot of additional continuous variables MIP B could also be extended to handle sequence-dependent setup-costs for Cases I and II, but again computational performance of MIP A is better despite of the larger number of binary variables. In Table 1 we recall the different objectives for the different cases.

Table 1. Objectives of the different cases

Minimization of	Case I	Case II	Case III	Case IV
setup-costs	•	•	-	-
earliness costs	•	-	•	-

2. Numerical example

A small instance illustrates the different models. 4 jobs labelled with 1..4 have to be scheduled on a single machine. Jobs 1,2 are from item D, jobs 3,4 from item E. Inventory holding costs h are translated into earliness costs. Between the jobs the setup-time and setup-cost matrices are given (with identical columns and rows for jobs from D and jobs from E, respectively). A setup "structure" of this kind can be motivated as item D having a light, E having a dark colour. Switching from light to dark is much less expensive and time consuming than a setup from dark to light. The first line gives the setup-times and setup-costs if job j is scheduled first.

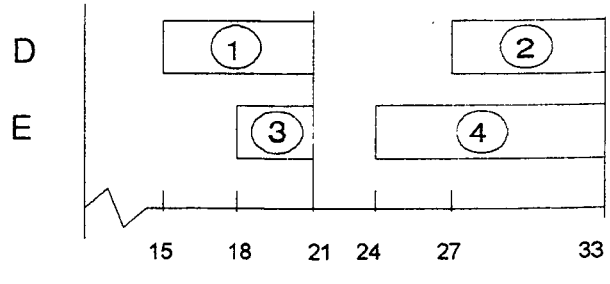
$$ST_{ij} = \begin{pmatrix} 3 & 3 & 2 & 2 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 3 & 3 & 0 & 0 \\ 3 & 3 & 0 & 0 \end{pmatrix} \qquad SC_{ij} = \begin{pmatrix} 90 & 90 & 60 & 60 \\ 0 & 0 & 30 & 30 \\ 0 & 0 & 30 & 30 \\ 90 & 90 & 0 & 0 \\ 90 & 90 & 0 & 0 \end{pmatrix}$$

For each job we know the earliness costs e_j, its deadline d_j and its processing time p_j (cf. Table 2).

Table 2. Data of the example

Item	h	j	e_j	d_j	p_j
D	1.5	1	9	21	6
		2	9	33	6
E	0.66	3	2	21	3
		4	6	33	9

Figure 1. Jobs scheduled at their deadline

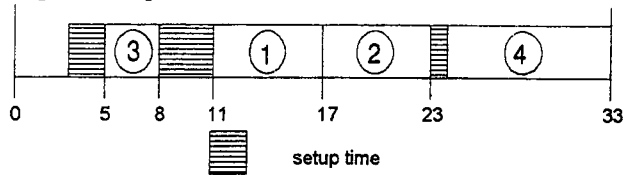


A solution can be represented by the decision variables sequence (S) and completion times (X). For Case I we get an optimal solution with costs of 60 + (21-8)·2 + 90 + (21-17)·9 + 0 + (33-23)·9 + 30 + 0 = 332 (Fig.2). Between jobs 1 and 2 we have a setup-time of 0, whereas between jobs 3 and 1 the setup-time is 3.

Table 3. Solution Case I

k	1	2	3	4
S_k	3	1	2	4
X_k	17	23	8	33

Figure 2. Optimal schedule Case I with 332



For the other Cases II (Fig.3) and III (Fig.4) different schedules are optimal. Note that idle time between jobs does not alter the setup state (Fig. 3).

Figure 3. Optimal schedule Case II with 120

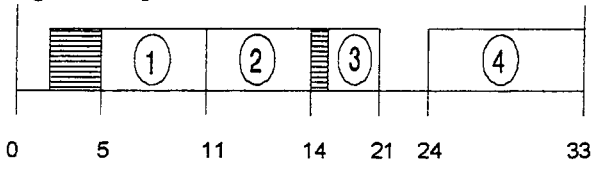
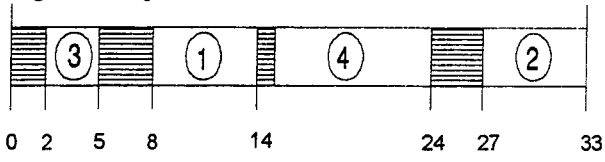


Figure 4. Optimal schedule Case III with 149



3. Computational Results

The conceptual model has been implemented in CHARME and the MIP formulation has been solved with OSL. We describe the experimental design and provide the solution times of 24 small instances solved for the four cases, so that 96 problems have been solved with both approaches. We consider 3 different "setup structures" A1, A2 and A3. Structure A1 has sequence-*independent*, A2 has sequence-*dependent* setup-times and -costs motivated through the colour example, costs correspond to setup-times. A3 has a setup-time matrix equal to A2, but setup-costs are arbitrarily. Instances have been generated as follows: For processing times and a setup-time matrix given a sequence is constructed, then deadlines are assigned to jobs so that the first, the second etc. job of each class have the same deadline. Thus lotsizing problems are imitated where demand often occurs in identical periods (for an integration of lotsizing and scheduling cf. Haase [7], Potts/Van Wassenhove [11]). Lower capacitated problems (L) are generated by multiplying deadlines with 1.4; problems with high capacity utilization (H) are supposed to have a small, problems with low capacity utilization a large solution space. Holding costs of each class are different, and average setup-costs approximately equal the holding costs incurred when two successive jobs are batched together. The instances are available from the authors upon request.

OSL runs under AIX Unix on an IBM RS 6000/550 RISC workstation, CHARME runs on an IBM 486/33Mhz PC under SCO Unix. The workstation is at least 2 times faster than the PC, factors differ from 2-5 depending on the operations (floating point or integer), so we took factor 2. Table 4 gives the

average running times of CHARME over 6 instances, i.e. the setup structures A1, A2, A3 and higher and lower capacity, respectively.

Table 4. Average CPU-times of CHARME in sec

#jobclasses	#jobs	Case I	Case II	Case III	Case IV
2	10	22	16	16	3
	14	109	87	75	12
3	9	46	33	33	2
	12	575	511	424	16

Apparently Case I is the most challenging problem where (as in lotsizing models) the sum of earliness (holding) and setup-costs has to be minimized.

We define *ratio* as

$$ratio = \frac{CPU-time \quad OSL}{CPU-time \quad CHARME} \cdot 2$$

which represents the different solution times of both solvers. The factor 2 takes into account the different machine speeds. In each entry of Table 5 we take the average *ratio* over the 3 different setup structures.

Table 5. ratio of CPU-times

#jobclasses	#jobs	capacity utilization	Case I	Case II	Case III	Case IV
2	10	H	4.6	109.6	0.2	0.2
		L	1.2	0.2	0.1	0.2
	14	H	a) 16.0	b) 75.0	0.2	0.4
		L	1.8	0.8	0.4	0.1
3	9	H	15.2	7.8	0.3	0.4
		L	1.0	0.4	0.4	0.6
	12	H	b) 5.6	c)	0.3	0.3
		L	0.8	0.2	0.4	0.5

a) 1 of 3 problems not solved by OSL
b) 2 of 3 problems not solved by OSL
c) none of 3 problems solved by OSL

The following facts merit attention:

Cases I and II:

With the formulation MIP A 8 of 96 problems were not solvable with OSL even in a very large amount of computation time. MIP solution times are widespread over the different setup structures, CHARME solution times differ much less. Solution times for CHARME are small for high capacity utilization (small solution space) but increase for loosely capacitated problems. In contrast OSL is slow for high capacity utilization (H) and becomes faster for lower utilization (L), so the ratios are high for "tight" problems.

Cases III and IV:

There solution times of OSL are generally better compared to CHARME. For Case III solution times of *both* solvers decrease with higher capacity utilization; for Case IV the reverse is true.

For all Cases I to IV CHARME solves problems with sequence-dependent setups (cf. structures A2, A3) much faster than those without sequence dependency. Especially the better performance of CHARME for tighter instances in Cases I and II is surprising.

4. Conclusions

A batch sequencing problem has been used to illustrate modelling and solving with general solvers, a MIP solver and a logic programming solver. For the batch sequencing problem modelling in CHARME, the logic solver, could be done much easier via a conceptual model taking decision variables as *indexes*. CHARME also does not need a matrix generator before solving a problem. To achieve a comparable performance, for different objectives different MIP-models had to be used, which required more modelling skill than for CHARME. High capacity utilization does not improve the LP-bounds for this sequencing problem, so that solution times of the MIP solver OSL are sometimes very large. Furthermore, modifications and extensions of the MIP formulation are more difficult.

For practical purposes it may also be interesting to get *all* solutions of a specific problem, a possibility easily offered by constraint propagation languages. Especially in an early stage of the modelling process this gives important insights (e.g. reveal forgotten constraints) and is also well suited for multicriteria decision making if other objectives have to be met.

In contrast to our results are experiences with CHARME when MIP formulations are implemented *directly* in CHARME. Then the branch-and-bound algorithm based on LP-relaxation solves the problem more effectively.

Acknowledgements:

We thank Ulrich Derigs from the University of Köln for helpful comments and Uwe Penke from the PC-Laboratory for his technical support.

References

[1] Baker, K.R., 1974. *Introduction to sequencing and scheduling*, Wiley, New York .

[2] Blazewicz, J., M. Dror and J. Weglarz, 1991. Mathematical programming formulations for machine scheduling: a survey, *European Journal of Operations Research,* Vol. 51, pp. 283-300.

[3] Bruno, J. and P. Downey, 1978. Complexity of task sequencing with deadlines, setup-times and changeover costs, *SIAM Journal on Computing,* Vol.7, pp. 393-404.

[4] BULL, 1990. Charme V1 User's Guide and reference manual, Artificial intelligence development centre, BULL S.A. France.

[5] Coleman, B. J., 1993. A simple model for optimizing the single machine early/tardy problem with sequence dependent setups, *Production and Operations Management*, Vol.1, pp. 225-228.

[6] Dhar, V. and N. Ranganathan, 1990. Integer programming versus expert systems: an experimental comparison, *Communications of the ACM*, Vol.33, pp. 323-336.

[7] Haase, K., 1993 . *Lotsizing and scheduling for production planning*, PhD Thesis, University of Kiel.

[8] Van Hentenryck, P., 1989. *Constraint Satisfaction in Logic Programming*, The MIT Press, Cambridge, MA, USA.

[9] IBM Corporation, 1992. Optimization Subroutine Library, Guide and reference, Release 2, Kingston NY, USA.

[10] Monma, C.L. and C.N. Potts, 1989. On the complexity of scheduling with batch setup times, *Operations Research*, Vol. 37, pp. 798-804.

[11] Potts, C.N. and L.N. van Wassenhove, 1992. Integrating scheduling with batching and lot-sizing: a review of algorithms and complexity, *Journal of the Operational Research Society*, Vol. 43, pp. 395-406.

[12] Sielken, R., 1976. Sequencing with setup costs by zero-one mixed integer linear programming, *AIIE Transactions*, Vol. 8, pp. 369-371.

[13] Unal, A. and A.S. Kiran, 1992. Batch sequencing, *IIE Transactions*, Vol. 24, pp. 73-83.

[14] Woodruff, D.L. and M.L.Spearman, 1992. Sequencing and batching for two classes of jobs with deadlines and setup times, *Production and Operations Management*, Vol.1, pp. 87-102.