Kimms, Alf

Optimal multi-level lot sizing and scheduling with dedicated machines

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 351

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

Suggested Citation: Kimms, Alf (1994) : Optimal multi-level lot sizing and scheduling with dedicated machines, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 351, Universität Kiel, Institut für Betriebswirtschaftslehre, Kiel

This Version is available at:
https://hdl.handle.net/10419/155423

Nr. 351

Optimal Multi-Level Lot Sizing and Scheduling with Dedicated Machines

A. Kimms

August 1994

Dipl.-Inform. Alf Kimms, Lehrstuhl für Produktion und Logistik, Institut für Betriebswirtschaftslehre, Christian-Albrechts-Universität zu Kiel, Olshausenstraße 40, 24118 Kiel, Germany

Abstract: This contribution deals with dynamic, capacitated, multi-level lot sizing and scheduling. The basic assumption is that for each item there is exactly one machine that this item can be manufactured on but no two items share a common machine. Finding an optimal solution with a greedy algorithm turns out to be an easy task.

Keywords: Production planning, lot sizing, scheduling, multi-level, dedicated machines, optimal greedy algorithm

1 Introduction

The integration of lot sizing and scheduling is approved by several authors [Drexl et al. 1994, Fleischmann 1990, Tempelmeier 1992] and the necessity of considering multi-level structures in production planning is commonly accepted [Afentakis and Gavish 1986, Drexl et al. 1993, El-Najdawi and Kleindorfer 1993, Kuik and Salomon 1990, Kuik et al. 1993, Maes et al. 1991, Stadtler 1994]. In spite of this, there is no contribution to capacitated, dynamic lot sizing and scheduling with general multi-level product structures that is known to us. To overcome this gap some research has recently be done and brought out several well-performing heuristic approaches for the single-machine case [Kimms 1993a, 1993b and 1994]. Now, a first step towards solving the multi-machine case is done by investigating the most simple situation: We assume that each item can be manufactured on exactly one machine and that no two items share a common machine. That is, that there is no competition for resources. We like to call such an environment a dedicated machine problem.

The following two chapters contain a precise definition of the problem and introduce a construction scheme that finds an optimal solution with low computational effort. Afterwards, computational studies show that the presented approach outperforms standard solvers.

2 Problem Definition

Informally, the problem under consideration can be stated as follows: Several items which are to be assembled with respect to a general (although acyclic) multi-level structure are to be produced within a finite time horizon that is subdivided into discrete periods of time. There is some known external demand for some of these items that may vary over time. To meet the demand for an item all its predecessing items are to be manufactured in advance where item-specific lead times (e.g. for cooling or transportation) are to be respected. The production of an item requires a certain amount of exactly one resource (e.g. a certain amount of time on a machine) where the available capacity of each resource is

scarce. The basic assumption here is that no two items require the same resource. Hence, setup costs and setup times are of no interest in this particular context and problems like splitting lots disappear. If items are produced much earlier than they are needed, they have to be stored in inventory which incurs holding costs. The objective is to find a cheap production plan that determines lot sizes and schedules.

Formally, this problem can be modeled as a linear program (LP):

$$\min \sum_{t=1}^{T} \sum_{j=1}^{J} h_j I_{jt} \tag{1}$$

subject to

$$I_{jt} = I_{j(t-1)} + q_{jt} - d_{jt} - \sum_{i \in S(j)} (a_{ji} q_{it}) \qquad (j = 1 .. J, t = 1 .. T) \tag{2}$$

$$I_{jt} \geq \sum_{\tau=t+1}^{\min\{t+v_j,T\}} \sum_{i \in S(j)} (a_{ji} q_{i\tau}) \qquad (j = 1 .. J, t = 0 .. T\text{-}1) \tag{3}$$

$$p_j q_{jt} \leq C_{jt} \qquad (j = 1 .. J (= M), t = 1 .. T) \tag{4}$$

$$I_{jt} \geq 0 \qquad (j = 1 .. J, t = 0 .. T) \tag{5}$$

$$q_{jt} \geq 0 \qquad (j = 1 .. J, t = 1 .. T) \tag{6}$$

where

a_{ji} is the „gozinto-factor", i.e. the quantity of item j that is needed to produce one item i;

C_{mt} is the capacity of the machine m in period t (for notational convenience we assume that item j is to be manufactured on machine j);

d_{jt} is the (external) demand for item j in period t;

h_j are the (non-negative) costs for holding one item j one period in inventory;

I_{jt} is the quantity of item j held in inventory at the end of period t (I_{j0} is the initial inventory);

J is the number of items;

M is the number of machines (M = J);

p_j is the amount of capacity consumed by producing one item j;

q_{jt} is the quantity of item j to be produced in period t;

$S(j)$ is the set of successors of item j, i.e. the set of items i where $a_{ji} > 0$;

T is the number of periods;

v_j is the (integral) lead time of item j ($v_j \geq 1$);

The objective to find a cheap production plan is written down in (1). (2) are the inventory balances. To make sure that lead times are respected, conditions (3) must hold. Restrictions (4) represent the capacity limits. (5) and (6) define decision variables to be non-negative.

For the sake of simplicity, we will furthermore assume no initial inventory. The integration of positive initial stock levels is discussed in [Kimms 1993a].

3 Computing Optimal Solutions

In contrast to the single-machine case [Kimms 1993a], the LP-formulation in chapter 2 contains neither integer nor binary variables. This is due to the assumption that there is no competition for resources and thus no setup state is to be considered. Standard solvers could therefore be employed to solve small and medium sized problems. But standard codes are inappropriate for large problems owing to large run-time. These problems for instance occur in fine-grain planning situations where the length of periods is „small" in relation to the overall time horizon (hence, the number of periods T is „large").

As we will see, there is an approach which finds an optimal solution with a run-time complexity of O (J×T). The clue to such an algorithm is that for minimizing holding costs a lot-for-lot production is the cheapest way to meet the demands. A proof will follow right after having the greedy algorithm introduced.

A backward oriented construction scheme fits well to solve multi-level problems [Kimms 1993a]: We start in period T and move stepwise towards period one. Let t be the period under consideration. The cumulative demand CD_{jt} for item j that is to be met is defined by

$$CD_{jt} = d_{jt} + \sum_{\tau=t+1}^{T} (d_{j\tau} - q_{j\tau})$$

where q_{jt} is set to zero for all $t = 1 .. T$ and all $j = 1 .. T$.

Since a lot-for-lot production defines the optimal solution, for each item j the following is now to be done: If the capacity of the machine that is dedicated to item j suffices to produce the cumulative demand, i.e. $C_{jt} \geq p_j \, CD_{jt}$, assign

$$q_{jt} = CD_{jt},$$

otherwise, assign

$$q_{jt} = \frac{C_{jt}}{p_j}.$$

No matter what case was true, update the demand matrix with

$$d_{i(t-v_i)} = d_{i(t-v_i)} + a_{ij}\, q_{jt}$$

for all those items i where $a_{ij} > 0$ holds. This updating neatly takes the multi-level structure into account. As long as no initial inventory is considered (as we do here), the instance is infeasible if $t\text{-}v_i < 1$ for at least one item i.

Period t-1 is considered next and the whole procedure loops until the schedule for period one is computed. Note, that the cumulative demand $CD_{j(t-1)}$ can easily be determined by evaluating the right hand side of the equation

$$CD_{j(t-1)} = d_{j(t-1)} + CD_{jt} - q_{jt}.$$

A final check for infeasibility can be made by testing whether or not there is any cumulative demand left which is not met.

An example will help to understand the working principle: Imagine a „gozinto"-structure with 4 items as depicted in Figure 1. Let all a_{ji}, v_j and p_j be equal to one. Assume a planing horizon of 6 periods of time. Demands and capacity limits per period are defined in Table 1. Table 2 contains the execution protocol of the greedy algorithm and Figure 2 the corresponding Gantt-chart.

Figure 1: The „gozinto"-structure of the example

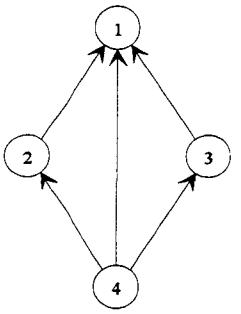


Table 1: External demands and capacity limits of the example

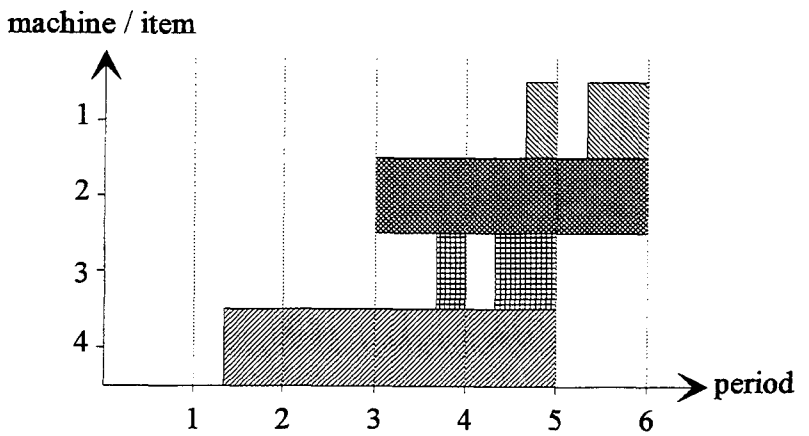
	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	capacity limit per period
j = 1					5	10	15
j = 2				10		20	15
j = 3							15
j = 4							20

Table 2: Execution protocol of the greedy algorithm when applied to the example instance

period t	cumulative demand CD_{jt}	production quantities q_{jt}	update operations
6	(10;20;0;0)	(10;15;0;0)	$d_{25} = d_{25} + 10 = 10$
			$d_{35} = d_{35} + 10 = 10$
			$d_{45} = d_{45} + 10 + 15 = 25$
5	(5;15;10;25)	(5;15;10;20)	$d_{24} = d_{24} + 5 = 15$
			$d_{34} = d_{34} + 5 = 5$
			$d_{44} = d_{44} + 5 + 15 + 10 = 30$
4	(0;15;5;35)	(0;15;5;20)	$d_{43} = d_{43} + 15 + 5 = 20$
3	(0;0;0;35)	(0;0;0;20)	
2	(0;0;0;15)	(0;0;0;15)	
1	(0;0;0;0)	(0;0;0;0)	

Some points of interest shall be discussed in more detail: When the procedure starts in period 6, the cumulative demands equal the external demands in period 6. Due to the capacity restrictions, the cumulative demand of 20 units cannot be satisfied in that period - only 15 units are scheduled. Because of the schedule in period 6, internal demands for items 2, 3 and 4 are to be met until period 5 which is taken into account by updating the demand matrix. In period 5 the cumulative demand for item 2 equals 15: 10 units were written into the demand matrix in the preceding step and 5 units are left over from the cumulative demand in period 6. Once again, the demand matrix is to be updated where the demand for item 2 in period 4 now sums up to 15: 5 units are the internal demand that is to be met for producing item 1 in period 5 and 10 units are the external demand for item 2 in period 4.

Figure 2: Gantt-chart of the solution



Since greedy algorithms in general do not necessarily result in optimal solutions [Korte et al. 1991, Nemhauser and Wolsey 1988], it shall now be proven that the greedy algorithm constructs optimal plans.

Theorem 1: Optimality of the greedy algorithm
If the problem instance is feasible then the greedy algorithm constructs an optimal production plan.
Proof: The backward oriented scheme by definition chooses to assign

$$q_{jt} = \min \{ CD_{jt} ; \frac{C_{jt}}{p_j} \}$$

for all j = 1 .. J if t is the current period. Hence, as long as shortages are not allowed, no production quantities can be shifted to the right. In other words, there is no feasible plan where items are produced later in time than scheduled by the greedy algorithm. So, if there exists a feasible plan then the algorithm is in the position to find it. Furthermore, due to the assumption of dedicated machines the plan where no shifts to the right are possible is a unique one.

What are the alternatives to this plan? Other feasible plans might be found by shifting some of the production quantities to the left.

In analogue to scheduling theory (for details we refer to [Baker 1974, Sprecher et al. 1993]) the constructed schedule can be termed an active schedule (although the notion of active schedules is usually defined on the basis of left shifts which is in contrast to what we do here). A well-known property of schedules with a so-called regular performance measure - like our objective function - is that an optimal schedule must be an active schedule of which there is only one in our case. As a consequence, the solution found with the greedy algorithm is an optimal solution. □

4 Computational Study

Since optimal solution approaches usually suffer from long run-time we shall be interested in whether or not our procedure improves the results reached by standard LP-solvers applied to the „easy" LP-formulation in chapter 2. The basis of our test-bed form 24 data instances with 5 items and 10 periods of time (namely, all class (a) problems defined in [Kimms 1993a]). These instances comprise different demand patterns and product structures which are those parameters that have the most significant influence on the shape of the simplex-tableau defined by the linear program. The details of these 24 instances are as follows: For each of 4 different product structures (see Figure 3) we defined 3 problems with external demand for end items only and 3 problems with external demand for all items.

The demand patterns were chosen as follows:

(1) External demand in period 10.

(2) External demand in periods 6 and 10.

(3) External demand in periods 6, 8 and 10.

Holding costs for item j are defined as $h_j = 6\text{-}j$. Lead times v_j and production coefficients a_{ji} are equal to one in all cases. The production of one item consumes one capacity unit while the capacity of each machine is assumed to be constant over time - we refer to Table 3 where capacity limits are given in dependence of demand patterns and product structures. The external demands per period (with respect to the demand pattern) are assumed to be 20 per item in the case that there is demand for end items only and 10 per item if there is external demand for all items.

Figure 3: Product structures

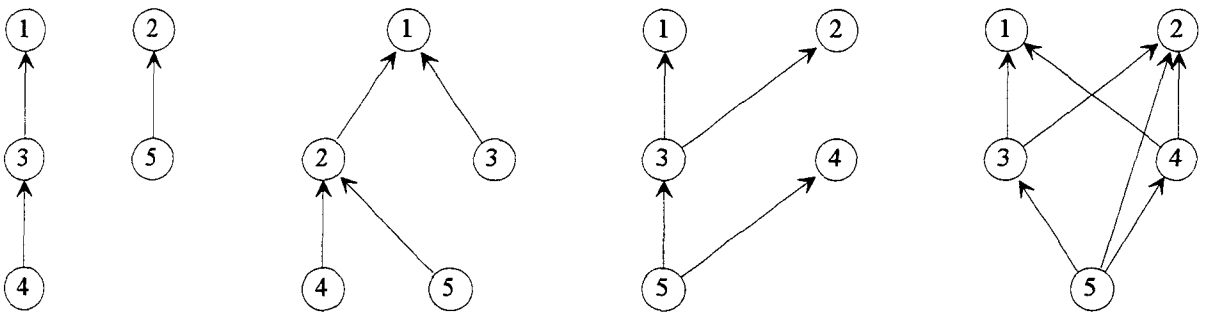


Table 3: Capacity constraints per period of time

	end items			all items		
	(1)	(2)	(3)	(1)	(2)	(3)
linear	35	35	100	35	35	100
assembly	35	35	100	35	35	100
divergent	35	100	100	35	100	100
general	35	100	200	100	100	200

A total of 6 different problem sizes (J × T) with 24 instances each are tested:

5 × 10		10 × 10
5 × 50	as well as	10 × 50
5 × 100		10 × 100

The 5 × 10 sized problems are the problems that form the basis of the test-bed where each of them has a feasible solution. The other problems were generated from these by performing copy

operations of the data which guarantees feasible solutions: E.g., if (▒) depicts the matrix that defines the „gozinto"-structure with 5 items then

$$\begin{pmatrix} \blacksquare & 0 \\ 0 & \blacksquare \end{pmatrix}$$

is the matrix that defines a „gozinto"-structure with 10 items. If (■) is the matrix of external demands with 5 items and 10 periods of time then

$$\begin{pmatrix} \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare \\ \blacksquare & \blacksquare & \blacksquare & \blacksquare & \blacksquare \end{pmatrix}$$

is the external demand matrix for 10 items and 50 periods of time. Analogously, if (▒) is a vector of a machine's capacity limits for 10 periods of time or a vector of holding costs, lead times or similar for 5 items then (▒▒) is the corresponding vector for a double number of periods or items, respectively.

The results in speed are as follows: Running on a 486 PC with 25 MHz, a C-implementation of our approach solves any instance of any of the above sizes within less than a second. Due to the construction scheme, the run-time of our approach increases linearly with the size of the problem. The standard LP-solver LINDO [Schrage 1991] needs up to several minutes on the same computer (see Figures 4 and 5).

Figure 4: Problem instances with 5 items

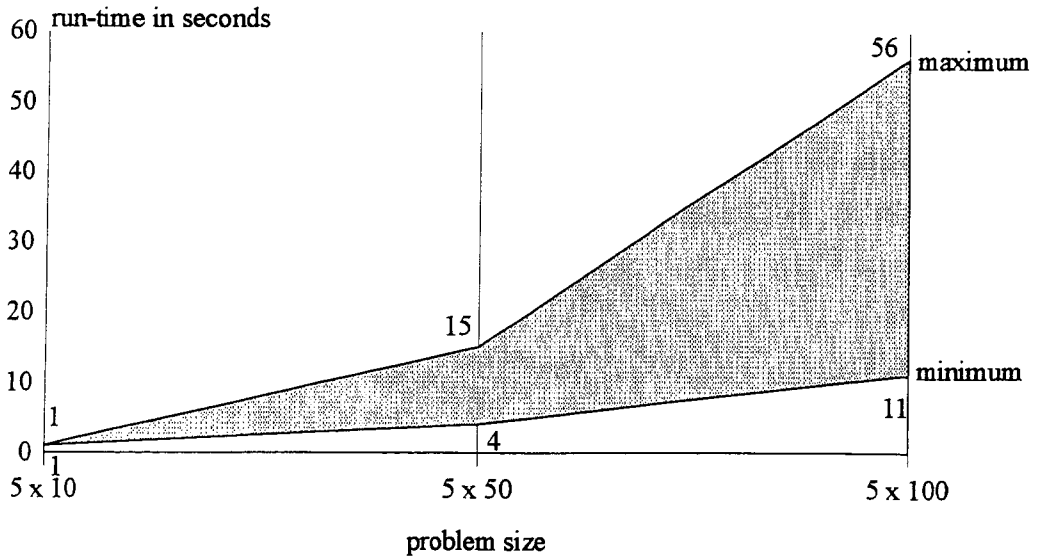
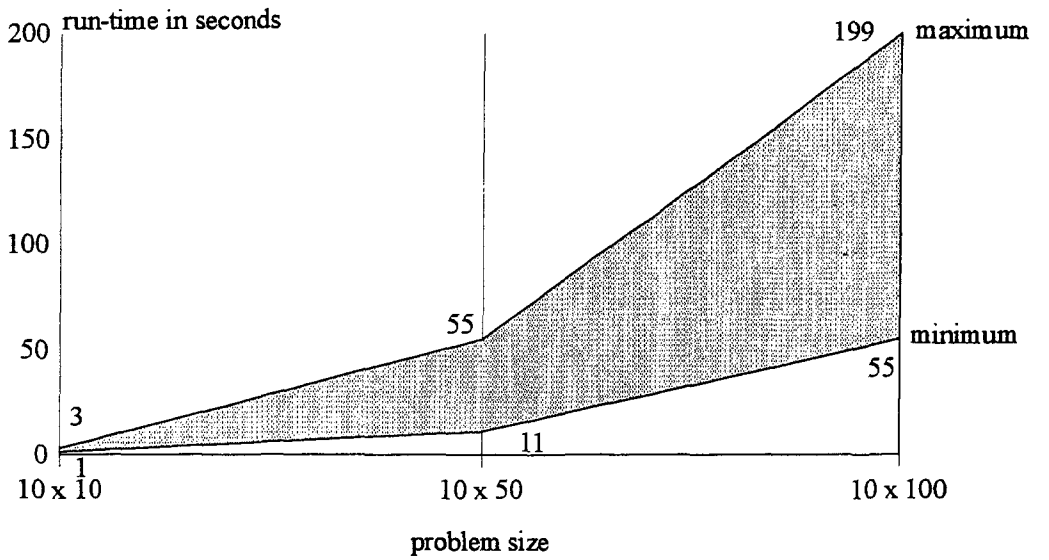


Figure 5: Problem instances with 10 items



Both figures show for each problem size the shortest and the longest time needed to solve one of the 24 instances. It is remarkable to note, that a larger problem size does not necessarily mean a longer run-time. E.g., one of the 5×100 sized problem instances was solved in 11 seconds while there exists a 5×50 sized problem instance that takes 15 seconds to be solved. Also note, that due to the copying by which we constructed the „gozinto"-structure of problems with 10 items the upper bound of the problems with 10 items tends to be underestimated. A more complex structure with 10 items is expected to cause longer run-times. Furthermore, the average run-time of the standard solver increases faster than linear which is in compliance with general statements [Bradley et al. 1977, Nemhauser and Wolsey 1988]. In summary, it is clear that our optimal approach outperforms standard solvers by orders of magnitude.

5 Conclusion

The focus of our attention was the dynamic, capacitated, multi-level lot sizing and scheduling problem. The basic assumption was that machines are dedicated to the items, i.e. no two items are produced on the same machine. On the one hand, an „easy-to-solve" LP-formulation was given. On the other hand a construction scheme was presented which computes optimal solutions by making use of problem insights. A computational study showed that the construction scheme works much faster than a standard solver which is applied to the linear program.

This contribution is a very first step towards multi-level lot sizing and scheduling with multiple machines. Future work must take situations into account where items share a common machine.

Acknowledgement

We are indebted to Andreas Drexl who always has one more good idea than we have.

References

Afentakis, P., Gavish, B., (1986), Optimal Lot-Sizing Algorithms for Complex Product Structures, Operations Research, Vol. 34, pp. 237-249

Baker, K. R., (1974), Introduction to sequencing and scheduling, New York, Wiley

Bradley, S. P., Hax, A. C., Magnanti, T. L., (1977), Applied Mathematical Programming, Reading, Addison-Wesley

Drexl, A., Haase, K., Kimms, A., (1993), Losgrößen- und Ablaufplanung in PPS-Systemen auf der Basis randomisierter Opportunitätskosten, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 333, Zeitschrift für Betriebswirtschaft, to appear

Drexl, A., Fleischmann, B., Günther, H.-O., Stadtler, H., Tempelmeier, H., (1994), Konzeptionelle Grundlagen kapazitätsorientierter PPS-Systeme, Zeitschrift für betriebswirtschaftliche Forschung, to appear

El-Najdawi, M. K., Kleindorfer, P. R., (1993), Common Cycle Lot-Size Scheduling for Multi-Product, Multi-Stage Production, Management Science, Vol. 39, pp. 872-885

Fleischmann, B., (1990), The Discrete Lot Sizing and Scheduling Problem, European Journal of Operational Research, Vol. 44, pp. 337-348

Kimms, A., (1993a), Multi-Level, Single-Machine Lot Sizing and Scheduling (with Initial Inventory), Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 329, European Journal of Operational Research, to appear

Kimms, A., (1993b), A Cellular Automaton Based Heuristic for Multi-Level Lot Sizing and Scheduling, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 331

Kimms, A., (1994), Complementary, Competitive Heuristic Approaches for Multi-Level Lot Sizing and Scheduling: Tabu Search and Randomized Regrets, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 348

Korte, B., Lovász, L., Schrader, R., (1991), Greedoids, Algorithms and Combinatorics, Vol. 4, Berlin, Springer

Kuik, R., Salomon, M., (1990), Multi-Level Lot-Sizing Problem: Evaluation of a Simulated-Annealing Heuristic, European Journal of Operational Research, Vol. 45, pp. 25-37

Kuik, R., Salomon, M., van Wassenhove, L. N., Maes, J., (1993), Linear Programming, Simulated Annealing and Tabu Search Heuristics for Lotsizing in Bottleneck Assembly Systems, IIE Transactions, Vol. 25, No. 1, pp. 62-72

Maes, J., McClain, J. O., van Wassenhove, L. N., (1991), Multilevel Capacitated Lotsizing Complexity and LP-Based Heuristics, European Journal of Operational Research, Vol. 53, pp. 131-148

Nemhauser, G. L., Wolsey, L. A., (1988), Integer and Combinatorial Optimization, New York, Wiley

Schrage, L., (1991), LINDO User's Manual for Linear, Integer and Quadratic Programming Release 5.0, Scientific Press

Sprecher, A., Kolisch, R., Drexl, A., (1993), Semi-Active, Active and Non-Delay Schedules for the Resource-Constrained Project Scheduling Problem, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 307, European Journal of Operational Research, to appear

Stadtler, H., (1994), Mixed Integer Programming Model Formulations for Dynamic Multi-Item Multi-Level Capacitated Lotsizing, Working Paper, Technical University of Darmstadt

Tempelmeier, H., (1992), Material-Logistik – Grundlagen der Bedarfs- und Losgrößenplanung in PPS-Systemen, Berlin, Springer, 2. Auflage