ECONSTOR
Make Your Publications Visible.

A Service of

ZBW Leibniz-Informationszentrum Wirtschaft
Leibniz Information Centre for Economics

Kimms, Alf

Working Paper — Digitized Version

Complementary, competitive methods for multi-level lot sizing and scheduling: Tabu search and randomized regrets

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 348

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

Suggested Citation: Kimms, Alf (1994) : Complementary, competitive methods for multi-level lot sizing and scheduling: Tabu search and randomized regrets, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 348, Universität Kiel, Institut für Betriebswirtschaftslehre, Kiel

This Version is available at:
https://hdl.handle.net/10419/155420

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. 348

Complementary, Competitive Methods for Multi-Level Lot Sizing and Scheduling: Tabu Search and Randomized Regrets

A. Kimms

July 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 presents two heuristic approaches for multi-level, single-machine lot sizing and scheduling. The first one is a variant of a so-called randomized regret based heuristic which is assumed to be the fastest available method for this particular class of problems. The second approach is a tabu search technique that is competitive with respect to both the run-time performance and the average deviation from the optimum objective function values. A computational study shows that these two methods are complementary, i.e. the randomized regret based heuristic performs good in those cases in which the tabu search gives poor results and vice versa. The combination of both thus provides a high performance approach to attack multi-level lot sizing and scheduling problems. It turns out that the average deviation from the optimum objective function value is round about 50% off from the stand-alone procedure results.

Keywords: Production planning, lot sizing, scheduling, PLSP, multi-level, randomized regrets, tabu search, heuristics

1 Introduction

Multi-level lot sizing and scheduling probably belongs to present days most toughest short-term production planning problems as indicated by the very small number of publications dealing with multi-level problems: [Afentakis and Gavish 1986, Bahl et al. 1987, Brüggemann and Jahnke 1994, Domschke et al. 1993, El-Najdawi and Kleindorfer 1993, Helber 1993, Iyogun and Atkins 1993, Kuik and Salomon 1990, Kuik et al. 1993, Maes et al. 1991, Roll and Karni 1991, Roundy 1993, Salomon et al. 1993, Stadtler 1994, Tempelmeier 1992, Tempelmeier and Derstroff 1993] is a non-comprehensive overview of this subject. Most of these references consider the lot sizing aspect only while scheduling, i.e. deciding in what sequence the lots of items are to be produced on a machine, is neglected. The literature which takes both aspects into account most often either imposes certain constraints on the product structure (e.g. assembly structures only or two levels only) or simplifies the problem by disregarding capacity limits. Some other publications assume demands to be constant over time and pay attention to an infinite planning horizon (as it stems from the EOQ-model [Andler 1929, Harris 1913]).

The problem that we consider is an extension of the single-level proportional lot sizing and scheduling problem (PLSP) [Drexl and Haase 1992, Haase 1994]: Within a finite planning horizon which is subdivided into discrete periods of time we have to produce several items on one single machine. The available machine capacity per period of time is limited while producing an item consumes an item specific amount of the capacity. External demands are assumed to be deterministic but dynamic and no shortages are allowed. Before an item can be produced to meet a demand all its predecessing items must be manufactured with respect to an item specific lead time. The production of these items may in turn cause (internal) demand for predecessing items and so on. We impose no other restriction on the product structure than being acyclic. Since the capacity of the machine is finite it happens that items are to be produced much earlier than they are needed to meet a demand. These items

are then to be held in inventory which incurs holding costs for any period of time that an item is stored. Usually a machine is in a certain setup state, so an item can only be produced if the machine is setup for this item. We stipulate that at most one changeover is allowed in each period of time. For setting a machine up setup costs are charged. The objective now is to find lot sizes and schedules which define a cheap production plan, i.e. a plan where the sum of setup and holding costs is as low as possible.

Solution methods for the above kind of problems are presented in [Kimms 1993a and 1993b] and their relevance when imbedded in the Manufacturing Resource Planning (MRP II) context is proved in [Drexl et al. 1993].

What follows is a mixed-integer program that gives a precise description of the problem under consideration. Afterwards, we introduce a sampling method based on randomized regrets which slightly improves recent bounds. Tabu search is then tailored to multi-level lot sizing and scheduling. A computational study shows that both techniques are competitive with respect to run-time performance as well as solution quality and that these methods are complementary. Concluding remarks finish the paper.

2 Mixed-Integer Program for Proportional Lot Sizing and Scheduling

The following mixed-integer program precisely models proportional lot sizing and scheduling problems [Kimms 1993a]:

$$\min \sum_{t=1}^{T} \sum_{j=1}^{J} \left(s_j x_{jt} + h_j I_{jt} \right) \tag{1}$$

subject to

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

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

$$\sum_{j=1}^{J} y_{jt} \leq 1 \qquad (t = 1 .. T) \tag{4}$$

$$B \left(y_{jt} + y_{j(t-1)} \right) - q_{jt} \geq 0 \qquad (j = 1 .. J, t = 1 .. T) \tag{5}$$

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

$$\sum_{j=1}^{J} (p_j q_{jt}) \leq C_t \qquad (t = 1 .. T) \qquad (7)$$

$$y_{jt} \in \{ 0, 1 \} \qquad (j = 1 .. J, t = 1 .. T) \qquad (8)$$

$$I_{jt} \geq 0, q_{jt} \geq 0, x_{jt} \geq 0 \qquad (j = 1 .. J, t = 1 .. T) \qquad (9)$$

where

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

B is a large number greater than $\frac{\max \{ C_t \mid t = 1 .. T \}}{\min \{ p_j \mid j = 1 .. J \}}$,

C_t is the capacity of the machine in period t;

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;

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 are the (non-negative) setup costs for item j;

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 ($v_j \geq 1$);

x_{jt} is a (binary) variable indicating whether a setup for item j occurs in period t ($x_{jt} = 1$) or not ($x_{jt} = 0$);

y_{jt} is a binary variable indicating whether the machine is setup for item j at the end of period t ($y_{jt} = 1$) or not ($y_{jt} = 0$) (y_{j0} is the initial setup state).

The objective (1) is to minimize the sum of setup and holding costs. The inventory balances are given in (2). Restrictions (3) make sure that internal demands are satisfied promptly with respect to item specific lead times. Constraints (4) define the setup state of the machine to be unique at the end of each period (and hence at the beginning of each period). The inequalities (5) must hold to guarantee that an item may be produced only if the machine is setup for this item. Those periods in which a changeover takes place (which is the case when the setup state of the machine for an item switches from zero to one) are spotted by (6). To integrate capacity constraints into the model, the conditions (7) are added. (8) defines setup state variables to be binary. Other decision variables are non-negative as it is expressed in (9). Note, that the setup variables x_{jt} are indeed zero-one valued since we face a minimization objective and the setup costs are non-negative.

In the sequel we will assume the initial inventory to be zero, i.e. $I_{j0} = 0$ for all items j (we refer to [Kimms 1993a] where it is shown that this assumption is restrictive in the multi-level case and where

the integration of positive initial inventories is discussed in detail). The initial setup state y_{j0} is considered to be uniquely given.

3 Heuristic No. 1: Randomized Regrets

The heuristic presented in this chapter refines a former one [Kimms 1993a] which in turn is a non-trivial multi-level extension of the single-level case [Haase 1994]. It belongs to the category of sampling methods because the basic idea is to generate (a large number of) different production plans from which the feasible plan with the lowest objective function value is chosen. As will be seen, there is no „history sensitivity". This is to say, that each production plan is generated without using any information obtained from previous plans. The construction scheme for a production plan shall now be described:

Starting off in period T with the matrix of (external) demands at hand, we move backwards to period one in a stepwise manner while scheduling at most two items within each period with respect to the fact that more than one changeover must not be done per period. Let t be the period under consideration. An item is scheduled in period t by a random choice among all items j with a positive entry d_{jt} in the demand matrix. The key to a „good" choice is a priority rule on the basis of which the probability to select an item is defined. For the sake of simplicity we delay the discussion of this rule until soon. Suppose now, that we have selected an item j in some way. Since we face capacity constraints, the quantity of item j that can be produced in t is

$$q_{jt} = d_{jt}$$

if the capacity in period t suffices the production of the whole demand, i.e. $C_t \geq p_j d_{jt}$, and

$$q_{jt} = \frac{C_t}{p_j}$$

if the capacity is exhausted, i.e. $C_t < p_j d_{jt}$. In the former case a second item can be scheduled likewise having $C_t - p_j q_{jt}$ capacity units available if either less than two items are scheduled in period t+1 or the item j is also scheduled at the beginning of period t+1 (if t = 1 the initial setup state must of course be respected and if t = T a second item can always be scheduled). In the latter case where item j needs all capacity units that are available in t, we simply update the demand matrix with

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

where the plan is infeasible if t-1 < 1 (this only holds if we consider no initial inventory as we do here). In both cases the demand matrix has to be updated with

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

for all items i where $a_{ij} > 0$ which neatly allows to handle internal demands uniformly to external demands. Again, the plan is infeasible if the time horizon is exceeded, i.e. if $t-v_i < 1$.

Now, a step is made to period t-1 and the whole process is repeated over and over again until period one is considered.

Noteworthy to say, that this scheme allows lots of items to be splitted since once that an item j is chosen in period t the production of the lot of size d_{jt} is preemptive. This is the case because due to the random choice of items at the beginning of every period the production is not necessarily continued in

period t-1 if the capacity in period t is exhausted. The simple example in appendix A shows that splitting lots may lead to better production plans.

What is left is the definition of a probabilistic decision rule for choosing an item out of the set of items with a positive demand. The rule that we use estimates the item specific regret not to schedule an item in the current period. The higher the estimated regret of an item the more probable is this item chosen. Roughly speaking, we have two aspects to take into account. On the one hand, we look for production plans where the sum of setup and holding costs is as low as possible. On the other hand, scarce capacities make life hard because even to find a feasible but suboptimal production plan is at least NP-complete [Salomon 1991].

Four criteria mirror these aspects (in our opinion). The sum of setup and holding costs is influenced by two facts:

(a) If an item is not scheduled in period t additional holding costs are incurred.

(b) If an item is not scheduled in period t one may save setup costs.

Before an item can be manufactured, all its predecessing items must have been produced. Not to produce an item in period t may thus lead to infeasibility because of two reasons:

(c) If an item is not scheduled in period t the depth of the product structure may exceed the time horizon.

(d) If an item is not scheduled in period t the available capacity may not be sufficient.

Paying attention to all the above criteria seems to be promising for defining a regret measure on the basis of which the selection of items leads to feasible and cheap production plans. A formal definition of such a regret measure can be given after having introduced some auxiliary notation:

Let CD_{jt} be the cumulated demand for item j which equals the item j's entry in the demand matrix when period t is considered. A path to item j (denoted as P_j) is a sequence of items $i_0 \rightarrow i_1 \rightarrow \ldots \rightarrow i_k \rightarrow \ldots \rightarrow i_K$ with $i_K = j$, $a_{i_{(k-1)}i_k} > 0$ for all $k \in \{1,\ldots,K\}$ and $a_{hi_0} = 0$ for all items $h \in \{1,\ldots,J\}$. The depth of an item j (denoted as dep_j) can now be defined as

$$dep_j = \max \{ \sum_{k=0}^{K-1} v_{i_k} \mid i_0 \rightarrow i_1 \rightarrow \ldots \rightarrow i_k \rightarrow \ldots \rightarrow i_K \text{ is a path to item j} \}.$$

To produce one item j we need to meet the internal demand for all predecessing items. The internal demand for item i to be satisfied for one item j (denoted as id_{ji}) is $id_{jj} := 1$,

$$id_{ji} = \sum_{h \in S(i)} (a_{ih}\, id_{jh})$$

if $i \neq j$ and there exists a path P_j such that $i \in P_j$ (where $i \in P_j$ is used to denote that item i is on path P_j), and $id_{ji} := 0$ if there is no such path P_j. The expression

$$cap_j = \sum_{i=1}^{J} (p_i\, id_{ji})$$

gives the capacity that is needed to produce one item j and all its predecessors and

$$AC_t = \sum_{\tau=1}^{t} C_\tau$$

is the total available capacity up to period t. Furthermore, let y denote the number of the item that was scheduled at the beginning of period t+1 when the focus is on period t (let y = 0 in period T).

The regret not to select item j when period t is considered is then specified as follows:

Case 1: $CD_{jt} > 0$ and $j \neq y$.

$$r_{jt} = \gamma_{1t} \frac{h_j \, CD_{jt}}{\max \{ s_i \mid \text{all items } i \}} \quad \text{(a)}$$

$$- \gamma_{2t} \frac{s_j}{\max \{ s_i \mid \text{all items } i \}} \quad \text{(b)}$$

$$+ \gamma_{3t} \frac{dep_j}{t - dep_j} \quad \text{(c)}$$

$$+ \gamma_{4t} \frac{CD_{jt} \, cap_j}{AC_t} \quad \text{(d)}$$

Case 2: $CD_{jt} > 0$ and $j = y$.

Just drop (b) in case 1.

Case 3: $CD_{jt} = 0$.

$r_{jt} = -\infty$

The subexpressions (a)-(d) directly correspond to the four criteria mentioned above. The influence of each of the criteria is controlled by the real-valued parameters $\gamma_{1t},\ldots,\gamma_{4t}$ where without loss of generality $0 \leq \gamma_{1t},\ldots,\gamma_{4t} \leq 1$ and $\sum_{i=1}^{4} \gamma_{it} = 1$ holds. The index t indicates that the relative relevance of the criteria may differ from period to period. To gain a finer control on the regret measure that is used to choose an item for production we introduce a modified regret measure:

$$\rho_{jt} = \begin{cases} 0 & \text{if } r_{jt} = -\infty \\ (r_{jt} - \min \{ r_{it} \mid \text{items } i \text{ with } CD_{it} > 0 \} + \varepsilon)^{\delta} & \text{otherwise} \end{cases}$$

The real-valued offset ε is a small value (e.g. $0.0001 \leq \varepsilon \leq 0.1$) which makes sure that a positive regret measure is assigned to every item with a positive cumulated demand. The positive, real-valued exponent δ amplifies ($\delta > 1$) or smoothes ($0 \leq \delta < 1$) the differences of the (unmodified) regrets.

As stated above, this procedure is a modification of a former one [Kimms 1993a]. There are two differences: First, we allow lot splitting now which was proved to be advantageous in some cases. And second, we choose a new set of the parameters $\gamma_{1t},\ldots,\gamma_{4t}$ in every period while in our former approach these parameters were kept time invariant.

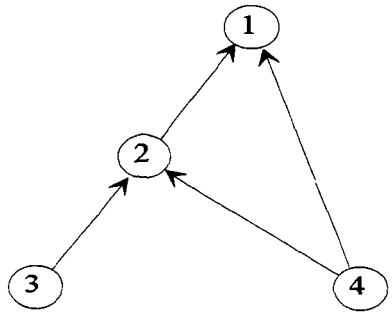
4 Heuristic No. 2: Tabu Search

Tabu search techniques are in a widespread use for solving combinatorial optimization problems (e.g. [Dell'Amico and Trubian 1993, Faigle and Kern 1992, Glover 1989 and 1990, Hertz and de Werra 1990, Nowicki and Smutnicki 1993]) and amaze with fast execution times and good suboptimal results. [Glover 1994] gives a nice overall view of the state-of-the-art in tabu searching. These techniques belong to the class of local search methods and thus follow the basic working principle which is common to all such approaches. This is to say, that starting with an initial feasible solution

transformations (which are to be defined with respect to the kind of problem) are performed to move from one feasible solution to another in a stepwise manner. The set of feasible solutions that can be reached from the current solution, say sol, by applying exactly one transformation step is called the neighborhood of sol and is denoted as N(sol). T(sol) denotes the set of transformations applicable to sol, i.e. the set of transformations that lead from solution sol to a solution sol′ ∈ N(sol). Associated with each feasible solution sol′ ∈ N(sol) is a value v(sol′) ∈ ℝ such that solution sol′ is assumed to be better than a solution sol″ ∈ N(sol) if v(sol′) < v(sol″) (for minimization problems v(sol) is often defined as the objective function value straightforwardly). We choose to move from solution sol to solution sol′∈ N(sol) if v(sol′) = min {v(sol″) | sol″ ∈ N(sol)} (if there is no unique best neighbor we can feel free to choose any of the best ones). So, since a move to a neighbor sol′ is done by following a steepest descent, mildest ascent strategy the neighbor may be worse than the current solution sol, i.e. v(sol) < v(sol′), and one may be trapped into a local optimum because of cycling. The fundamental idea now that coins the name tabu search is to keep a list of solutions that are forbidden to be visited. The neighbors that are allowed to be the destination of a step from solution sol are those neighbors of sol only that are not listed in the tabu list. Once that a step from solution sol to solution sol′ is done, sol is added to the list of tabu solutions to prevent cycling (if tabu lists are of finite length as they usually are cycling cannot be prevented for on long-term). Some authors use a list of transformations instead where once that a move from solution sol to solution sol′ is made, the reverse transformation that leads from sol′ to sol is appended to the list. The size of the tabu list may either be fixed or variable, i.e. tabu lists may be of static or dynamic length where old entries are deleted when new entries are added to keep the length of the list in size. Because tabu lists happen to be too restrictive, a boolean so-called aspiration function is introduced that works as an ultimate decision maker to decide whether or not a solution, though being under a tabu by the tabu list, is actually not acceptable.

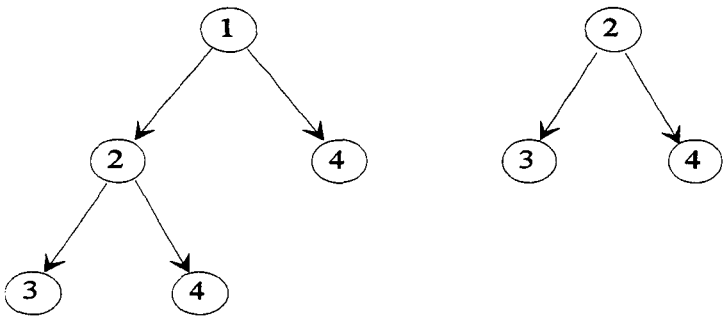
To apply the tabu search technique to multi-level, single-machine lot sizing and scheduling problems we exploit the idea of disjunctive arcs [Balas 1969]. Roughly speaking, each PLSP-instance is converted into a connected digraph where a node represents an external or internal demand for an item and the set of arcs defines a partial ordering in which the demands are to be satisfied. Elaborating an example will help to understand the details: Let us consider a product structure consisting of four items as given in Figure 1.

Figure 1: A product structure with four items



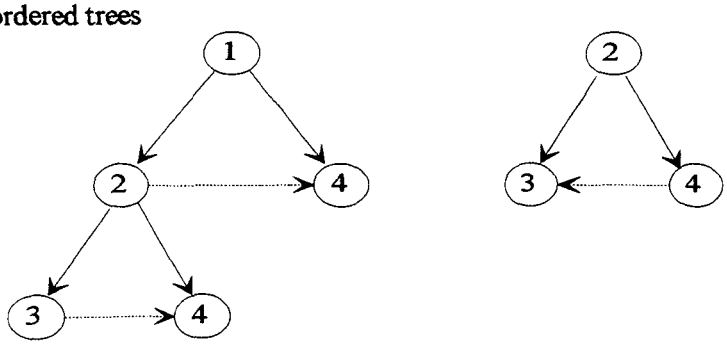
Before we turn to describe how to solve a particular PLSP-instance, we define some auxiliary notions. We start with the notion of a tree induced by a node j. Let Pre(j) denote the set of immediate predecessors (e.g. Pre(1) = {2, 4} with respect to Figure 1). Given an acyclic product structure, the tree that is induced by node j is the tree which is recursively constructed in the following way: A copy of node j builds the root of the tree and an arc points from node j to the root of (a copy of) every subtree induced by each node i ∈ Pre(j). Figure 2 shows the trees induced by nodes 1 and 2, respectively. By definition root nodes in an induced tree have no predecessors while any other node has exactly one predecessing node. The graph structure introduced right now will later on be used to maintain the information that before an item can be produced, a certain number of predecessing items must be manufactured in advance to meet the internal demand with respect to the multi-level product structure.

Figure 2: Trees induced by nodes 1 and 2, respectively



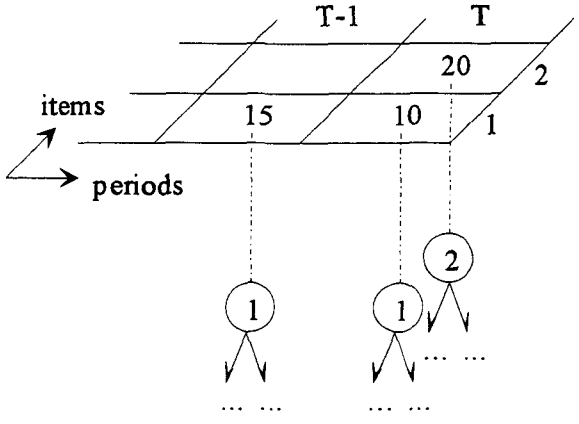
Furthermore, we will call a tree induced by a certain node partially ordered if there is a total order for each set of nodes which share the same predecessing node. Figure 3 displays the partially ordered trees of Figure 2 whereby a total order among nodes with the same predecessor is symbolized by dotted arcs. In general of course, for a set of nodes with the same predecessor there is more than one total order possible and hence a tree can be partially ordered in more than one way. It should be noted that a partially ordered tree is not a tree in the common sense [Bollobás 1979] anymore, since due to the arcs introduced by the total orders which are defined on subsets of the nodes, some nodes do now have more than one predecessor. However, we use the term partially ordered „tree" because it concisely describes how such a graph is constructed. Partially ordered trees will later on be used to support the decision process of sequencing the production of items.

Figure 3: Partially ordered trees



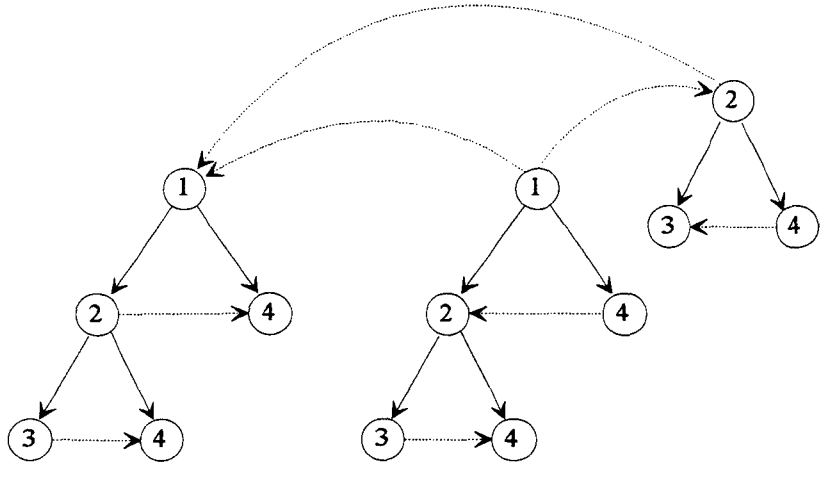
Coming back to the description of how to solve a particular PLSP-instance, we will now introduce the graph structure on the basis of which a tabu search is performed. Given a demand matrix containing the external demands of the instance under consideration, a connected digraph is constructed by first creating partially ordered trees for each positive entry in the demand matrix where each tree is induced by the node that corresponds to the item with the external demand (see Figure 4).

Figure 4: Correspondence between partially ordered trees and the external demand matrix



A total order is then defined between the root nodes of these trees by adding additional arcs (see Figure 5). These additional arcs will later on define the order in which external demands are satisfied. In summary we obtain a connected, acyclic digraph where the set of arcs can be subdivided into two disjoint sets: AG, the set of „gozinto"-arcs that originate from the product structure and AO, the set of arcs that define the orders in the partially ordered trees and among the root nodes.

Figure 5: A total order between the root nodes defines a connected, acyclic digraph



The basic working principle of the tabu search heuristic now consists of two phases which are alternatingly repeated. Starting with an initial graph structure, in a first step a production plan is derived that respects the orders as defined by the graph. In a second step the graph is modified by redirecting one (or some) of the arcs in the arc set AO in such a way that a connected, acyclic digraph is obtained

again. Another production plan is then constructed and so on. This is repeated over and over again for a large number of times, say, 1000 times or so. Afterwards, we simply choose the production plan with the lowest objective function value and we are done.

Before we will describe the way a graph structure is rewritten which is the part in which we employ a tabu search, we will focus on the construction of a production plan first. While the above explanations are sufficient to understand how a graph structure is defined, we certainly need some more information to construct a feasible production plan. That is, the nodes of the graph must not contain the number of an item only, we furthermore need to know the demand size that is represented by each node as well as the latest time that this demand is to be met. Thus a tuple of the following form is attached to each node:

[item number , demand size , deadline , reference count]

where the item number is as defined above. The demand size equals the corresponding entry of the demand matrix, i.e. the external demand in the case of a root node of a partially ordered tree and it equals the internal demand in all other cases. The internal demand, by the way, can easily be computed by

$$ds_{i_0} \cdot \prod_{k=1}^{k=K} a_{i_k i_{k-1}}$$

where i_0 is the item number in the root node of the actual partially ordered tree, i_K is the number of the item under consideration and $i_0 \rightarrow i_1 \rightarrow \ldots \rightarrow i_{K-1} \rightarrow i_K$ is the path from the root node to the node of interest. The factor ds_{i_0} denotes the demand size entry of the root node i_0.

The deadline entry of a node is t in the case of a root node that corresponds to a demand matrix entry in period t. For all other nodes the entry is undefined when we start the construction of a production plan and will be defined upon proceeding. Last, but not least, the reference counter contains the number of arcs which are members of the arc set AO and which point to the respective node. For instance, if we consider the graph structure in Figure 5 the reference count entry of the root node 1 that corresponds to the demand matrix entry in period T-1 is 2 and the reference count of the node 3 contained in the tree induced by item 2 is 1.

The scheme for constructing a production plan is a backward oriented one again since we know that this is appropriate for multi-level structures [Kimms 1993a]. While moving from period T to period one the overall construction rule is to consider a node, i.e. a demand for an item, only if all the nodes that are predecessors with respect to the partial order defined by the graph were considered before. To ensure this, we maintain a list of nodes with all the nodes in it that have no unconsidered predecessors with respect to the „gozinto"-arcs only. Initially, this list consists of all the root nodes and no more than just these. Suppose now, that period t is the focus of our attention. To decide which item, if any item at all, to schedule in period t, we make a pure (uniformly distributed) random choice between those nodes in the list which have a reference count equal to zero. Note, that there will be at least one such node if and only if there are unconsidered nodes left, otherwise we will have finished the construction. Let t' be the deadline entry of the chosen node and j be the corresponding item. Two situations may now appear: First, it may happen that $t' \geq t$ in which case item j is scheduled in period t. If the demand size exceeds

the capacity in period t we allocate additional capacity units in period t-1 and earlier if necessary which is different from what was said about the regret based heuristic. Second, it may happen that $t' < t$ which causes item j to be scheduled in period t' analogously and t' to be the new focus of attention. In the latter case the skipped periods are idle since we move strictly on towards period one and have no chance to reconsider periods later than the current one. In both cases we remove the selected node from the list and decrease the reference counting entries by one for all the nodes to which an arc of the arc set AO directly points to from the node just removed. Furthermore, we append to the list all those nodes to which an arc of the arc set AG directly points to and initialize the deadline entry of such a node with the number of the current period minus the lead time v_i of the corresponding item i. Afterwards, the node list is inspected in search for nodes with an item number equal to j, with a deadline greater than or equal to the current period and with a reference count equal to zero. Proceeding along the same lines, each of them is integrated into the production plan as well. Once that the scheduling of item j comes to an end, it may happen that the capacity of the current period is not exhausted. Similar to the regret based heuristic, a second item can than be chosen if it is guaranteed that at most one changeover is to be done in that period. Stepping one period ahead, this scheme is repeated. A feasible production plan is eventually found, if no node remains in the node list.

An example will help to make the way of construction transparent: Suppose a product structure as given in Figure 1 where „gozinto"-factors and lead times equal 1 in all cases. Let T = 10 and assume that the demand matrix is defined as in Figure 4 with three non-zero entries only. The graph structure that is derived from this data shall look like in Figure 5. Furthermore, let us consider a capacity limit of 25 units per period where one unit is consumed by producing one unit of an item's demand. Table 1 provides a possible execution protocol of the construction scheme. The resulting production plan can be found in Figure 6.

The protocol contains exactly as many steps as there are nodes in the connected, acyclic digraph since each step schedules one node. For each step the protocol shows how the list of nodes looks like before a node is chosen, at what period of time the focus of attention is, the node that is selected from those nodes in the list which have a reference count entry equal to zero and what change to the production plan eventually occurs. Nodes in the list that are adjacent with respect to the arc set AO are underscored with one continuous line. Some points of interest in this example should be discussed in more detail:

- Step 1: The list that we start with contains all root nodes.
- Step 1: Once that a node ([1,10,10,0]) is chosen, this node is removed from the list while all those nodes are added that are immediate successors of the node with respect to the arc set AG ([2,10,9,1] and [4,10,9,0]). The deadline entry of the added nodes (9) is computed by the number of the current period (10) minus the lead time for these items(1). The reference count entry of those nodes to which an arc of the arc set AO points to from the chosen node are decremented by one ([2,20,10,0] and [1,15,9,1]).

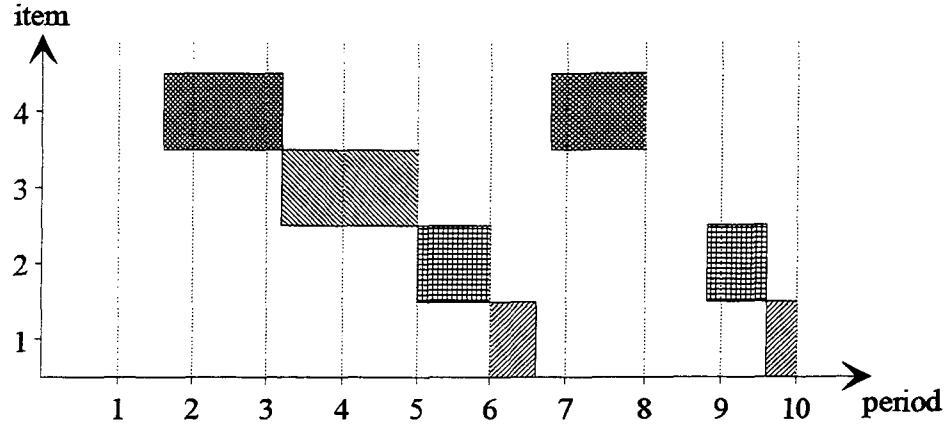
Table 1: A protocol of a construction phase

step	list of nodes	period	chosen node	quantities
1	[1,10,10,0] [2,20,10,1] [1,15,9,2]	10	[1,10,10,0]	$q_{1,10} = 10$
2	[2,10,9,1] [4,10,9,0] [2,20,10,0] [1,15,9,1]	10	[2,20,10,0]	$q_{2,10} = 15$
				$q_{2,9} = 5$
3	[2,10,9,1] [4,10,9,0] [3,20,8,1] [4,20,8,0] [1,15,9,0]	9	[4,20,8,0]	$q_{4,8} = 20$
4	[2,10,9,1] [4,10,9,0] [3,20,8,0] [1,15,9,0]	8	[4,10,9,0]	$q_{4,8} = 25$
				$q_{4,7} = 5$
5	[2,10,9,0] [3,20,8,0] [1,15,9,0]	7	[1,15,9,0]	$q_{1,7} = 15$
6	[2,10,9,0] [3,20,8,0] [2,15,6,0] [4,15,6,1]	6	[2,15,6,0]	$q_{2,6} = 15$
7	[2,10,9,0] [3,20,8,0] [3,15,5,0] [4,15,5,1] [4,15,6,0]	6	[2,10,9,0]	$q_{2,6} = 25$
8	[3,10,5,0] [4,10,5,1] [3,20,8,0] [3,15,5,0] [4,15,5,1] [4,15,6,0]	5	[3,20,8,0]	$q_{3,5} = 20$
9	[3,10,5,0] [4,10,5,1] [3,15,5,0] [4,15,5,1] [4,15,6,0]	5	[3,15,5,0]	$q_{3,5} = 25$
				$q_{3,4} = 10$
10	[3,10,5,0] [4,10,5,1] [4,15,5,0] [4,15,6,0]	4	[3,10,5,0]	$q_{3,4} = 20$
11	[4,10,5,0] [4,15,5,0] [4,15,6,0]	4	[4,15,5,0]	$q_{4,4} = 5$
				$q_{4,3} = 10$
12	[4,10,5,0] [4,15,6,0]	3	[4,10,5,0]	$q_{4,3} = 20$
13	[4,15,6,0]	3	[4,15,6,0]	$q_{4,3} = 25$
				$q_{4,2} = 10$

- Step 2: A second item (item 2) can be scheduled in period 10 since the first item (item 1) does not consume all the capacity units. For capacity being scarce, some of the demand units of the second item are the be scheduled in period 9 (lots are not splitted here).
- Step 3: A node is chosen that contains a deadline entry (equal to 8) which is less than the number of the current period (9). Hence, this item is scheduled in period 8 and period 9 is not considered any further.
- Step 4: Since a node with item number 4 was chosen in step 3 and there is another node with the same item number, with a deadline greater than or equal to the number of the current period and with a reference count equal to zero ([4,10,9,0]), this node is to be chosen next.

What is left yet is the description of the tabu search guided modification of the graph structure. The initial graph that we start with is constructed in the way described above. By arbitration we define the initial orientation of arcs which do not stem from the „gozinto"-structure as follows: An arc points from node j to node i if $j < i$ and if they share a common predecessor (this defines the partially ordered trees). Furthermore, an arc points from root node j to root node i which do correspond to a demand matrix ent. , in period t_j and t_i, respectively, if either $t_j > t_i$ (which is the first case) or $t_j = t_i$ and $j < i$ holds.

Figure 6: Gantt-chart for the production plan which is constructed as defined in Table 1



Given a connected, acyclic digraph we redirect one arc in the arc set AO to obtain an acyclic graph again. A variant that comes immediately to mind is the redirection of more than one arc, but this is not considered any further in this paper. Apparently, not all members of the arc set AO are valid candidates for the redirection operation since cycles are to be circumvented. An arc between two nodes, say from node j to node i, can be redirected without harm if and only if the reference count of node j is exactly one less than the reference count of node i. In all other cases, a cycle would be introduced. It is noteworthy to say, that at most one arc with this property originates from each node. Proofs are simple and can thus be omitted.

At least one open question still remains. That is, how to select an arc for redirection? Since the number of arcs which could be reversed is expected to be quite high in general, it seems to be not a good idea with respect to the run-time performance to consider all potential transformations which perhaps is the best way to detect a „good" transformation. Indeed, the large size of a neighborhood is one of the main reasons for which local search methods perform slow in some applications. To reduce the effort of finding a „good" arc that is to be reversed, we choose some valid candidates at random and decide to redirect the „best" one of those. The choice of these candidates is done as follows: In a first step, we randomly choose a certain number of nodes with a uniform distribution (some nodes may be chosen more than once, of course). In a second step, for each of these nodes we find the arc (if such arc exists) that originates from that node and that can be reversed without introducing a cycle. The job now is to select the „best" of these arcs. The terms of being „better" or „worse" when two arcs are compared could be defined on the basis of the objective function value. In this case, an arc would have to be temporarily reversed, a production plan would then have to be constructed (as described above) and the objective function value of this plan would have to be computed. An arc that leads to a plan with minimum objective function value would be considered to be the „best" and permanently reversed. This scheme obviously is very inefficient. Instead of comparing objective function values, we therefore use an estimate of the improvement of the objective function value. The arc with the best estimated improvement is then reversed. But how to estimate the influence of an arc on the construction of a production plan? The point of view that gives an answer to this question is the following: Let us

suppose, that the arc that we care about connects two nodes, say j and i. Without loss of generality let node i be the destination of the arc. Furthermore, let ds_j and ds_i denote the respective demand size entries of nodes j and i. If the orientation of this arc is reversed, item i must be scheduled before item j when a production plan is constructed. Since we employ a backward oriented scheme for construction, we expect to store some additional ds_j items j in inventory for at least one more period while we can expect to decrease the holding time of ds_i items i for at least one period. In summary, it makes sense to expect that the holding costs of item j increase and the holding costs of item i decrease. Setup costs may or may not change. So the estimated change of the objective function value is

$$h_j\, ds_j - h_i\, ds_i.$$

The lower this value, the more promising it is to reverse the considered arc. If this value is negative, one could expect the new production plan to improve the recent one. Of course, it may happen that no feasible production plan can be constructed after reversing the selected arc. But we accept this and allow transformations to be done though leading to infeasible solutions. This strategy could be termed „tunneling" which expresses (the hope) that while bypassing infeasible solutions a feasible solution is eventually reached again. To follow this strategy is advisable if the set of feasible solutions is disconnected, i.e. given a feasible solution, one cannot reach any other feasible solution by performing basic transformation steps with intermediate solutions that are also feasible. In our application this is in general the case. A simple example in appendix B proves this statement.

Until now, tabu search has played no role, but is introduced right now to avoid cycling. In this particular context, once that an arc from node j to node i is redirected, the reverse operation, i.e. redirecting the arc from node i to node j, is entered into the tabu list where the tabu list has a fixed length and is empty when we start. Following the guidelines of the tabu search scheme, an arc that is chosen at random to be considered, is now considered only if it is not contained in the tabu list. The only exception is that the estimated change of the objective function value is negative which functions as the aspiration criterion. If no arc is chosen to be redirected (which may happen for instance if all the arcs that are chosen at random are already contained in the tabu list) an empty entry is added to the tabu list instead to make sure that we do not get stuck while modifying the graph.

5 Computational Study

To study the presented heuristics we use a standard test-bed [Kimms 1993a and 1993b] which contains a collection of 144 samples with 5 items and 10 periods of time. The size of these problems was chosen because it is large enough to construct non-trivial problems and small enough to be solved with standard MIP-solvers within reasonable time which is necessary to get an idea of how good or bad a heuristic really is. Both heuristics were implemented in C running on a 486 PC with 25 MHz.

The details of the test data are as follows: Four different product structures are considered (see Figure 7). For each of these product structures we test six demand patterns where three of them define

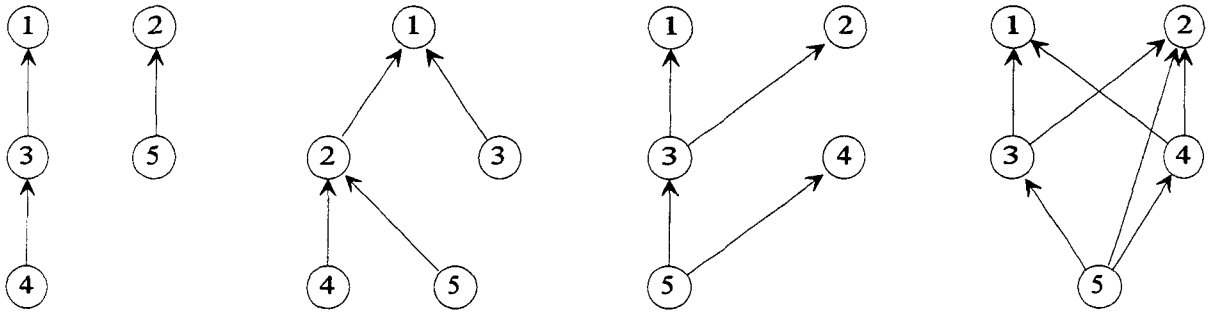
demand for all but only end items and three of them define demands for all items. The three basic demand patterns mainly differ in the way the demand matrix is filled with values:

(1) External demand in period 10.

(2) External demand in periods 6 and 10.

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

Figure 7: Product structures of the sample problems



To identify these problems we use a triple notation $\sigma/\pi/\nu$ where $\sigma \in \{L, A, D, G\}$ is a short notation for the product structure, $\pi \in \{E, A\}$ indicates whether demand occurs for end items or for all items, and $\nu \in \{1, 2, 3\}$ corresponds to one of the above demand patterns. These 24 data frames (4 product structures combined with 6 demand patterns each) are completed with 6 different data instances each which gives the total of 144 problems (feasible solutions do exist for each instance):

(a) Holding and setup costs are defined as in Table 2. Lead times and production coefficients are equal to one in all cases, i.e. $v_j = 1$ and $a_{ji} = 1$. The production of one item (of any kind) consumes one capacity unit, i.e. $p_j = 1$ for all items, while the capacity of the machine is assumed to be constant over time. More precisely, we assume the capacity constraints per time period to be defined as in Table 3. The external demands per period (with respect to the demand pattern ν) are assumed to be 20 per item in the case that $\pi = E$ and 10 per item in the case that $\pi = A$.

(b) The same data set as (a) except setup costs being multiplied by 20.

(c) The same data set as (a) except external demand sizes being half of what is defined above, i.e. 10 items if $\pi = E$ and 5 items if $\pi = A$.

(d) The same data set as (b) except external demand sizes being chosen as in (c).

(e) The same data set as (c) except $p_j = 0.5$ for all items and all production coefficients are doubled, i.e. $a_{ji} = 2$ for all items j and i with respect to the product structures defined above.

(f) The same data set as (d) except p_j and a_{ji} being chosen alike (e).

Table 2: Setup and holding costs

item	setup costs	holding costs
1	30	5
2	20	4
3	20	3
4	10	2
5	10	1

Table 3: Capacity constraints per period of time

	$\pi = E$			$\pi = A$		
	$\nu = 1$	$\nu = 2$	$\nu = 3$	$\nu = 1$	$\nu = 2$	$\nu = 3$
$\sigma = L$	35	35	100	35	35	100
$\sigma = A$	35	35	100	35	35	100
$\sigma = D$	35	100	100	35	100	100
$\sigma = G$	35	100	200	100	100	200

For both heuristics we provide the deviation from the optimum objective function value which is computed by

$$\text{deviation} := 100 \cdot \frac{F^*_H - F^*_{opt}}{F^*_{opt}}$$

where F^*_H denotes the result of the heuristic and F^*_{opt} denotes the optimum objective function value. The results given are computed after 1000 repetitions, i.e. for each problem instance a total of 1000 production plans are (tried to) constructed from which the best one is chosen. We start with the results of the randomized regret based heuristic (Tables 4 to 6). The parameters of this heuristic are chosen as follows: ε and δ are chosen at random from the intervals [0.0001 , 0.1] and [0 , 10], respectively, and $\gamma_{1t}, \ldots, \gamma_{4t}$ are chosen at random from the interval [0 , 1]. To perform 1000 repetitions the randomized regret based heuristic needs between 4 and 9 seconds per problem instance.

Table 4: Deviation of the randomized regret based heuristic, $\nu = 1$

	(a)	(b)	(c)	(d)	(e)	(f)	average
L/E/1	0.00	0.00	0.00	0.00	0.00	0.00	0.00
L/A/1	10.34	1.50	7.89	0.79	5.88	0.76	4.53
A/E/1	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A/A/1	12.86	5.83	6.82	0.00	4.48	0.73	5.12
D/E/1	10.23	2.09	8.33	1.03	5.13	0.46	4.54
D/A/1	19.05	2.93	14.63	1.57	10.91	0.50	8.26
G/E/1	7.75	9.50	0.00	0.00	2.19	2.43	3.64
G/A/1	8.89	17.36	7.41	1.52	3.49	2.80	6.91
average	8.64	4.90	5.64	0.61	4.01	0.96	4.13

Table 5: Deviation of the randomized regret based heuristic, $\nu = 2$

	(a)	(b)	(c)	(d)	(e)	(f)	average
L/E/2	8.55	33.09	11.11	49.09	6.67	40.40	24.82
L/A/2	12.70	24.10	14.10	33.26	10.68	29.38	20.70
A/E/2	2.80	39.93	2.33	60.18	3.61	51.32	26.69
A/A/2	10.00	27.46	11.70	31.42	16.31	35.52	22.07
D/E/2	12.36	24.20	11.32	32.11	18.29	27.05	20.89
D/A/2	23.88	28.06	20.24	31.88	18.42	22.24	24.12
G/E/2	0.00	17.51	0.00	29.77	12.21	27.68	14.53
G/A/2	15.63	23.58	16.22	28.86	23.53	27.58	22.57
average	10.74	27.24	10.88	37.07	13.72	32.65	22.05

Table 6: Deviation of the randomized regret based heuristic, $\nu = 3$

	(a)	(b)	(c)	(d)	(e)	(f)	average
L/E/3	12.37	26.67	10.71	37.50	5.62	33.10	20.99
L/A/3	17.78	15.06	18.10	15.85	13.48	14.47	15.79
A/E/3	19.27	39.74	11.11	49.79	9.38	48.00	29.55
A/A/3	16.95	16.47	12.59	13.31	11.17	16.75	14.54
D/E/3	13.91	20.00	13.10	22.94	21.60	16.38	17.99
D/A/3	23.30	13.86	19.17	16.96	21.69	12.54	17.92
G/E/3	3.17	15.14	6.03	27.06	18.63	24.50	15.76
G/A/3	17.69	14.14	19.63	15.41	24.51	13.91	17.55
average	15.55	20.13	13.81	24.85	15.76	22.46	18.76

Applying the tabu search heuristic to the same set of problems gives the results provided in Tables 7 to 9. The parameters that control the heuristic are chosen as follows: The size of the tabu list has a fixed length of five entries (where it is noteworthy to be mentioned that the length of the list seems to be not very important in this context). To find an arc that is to be reversed, 20 nodes are chosen at random. Performing 1000 repetitions of the tabu search heuristic takes between 4 and 7 seconds per problem.

When both heuristics are compared, one can see that they are competitive with respect to the run-time and the average deviation from the optimum objective function value. A closer look brings out, that both heuristics are complementary as well. The randomized regret based heuristic performs well in the case of problem categories (a), (c) and (e) in which setup costs are quite low. Building lots is not the key to cheap production plans for these problems. If setup costs are considerably large the randomized regret based heuristic is not that good. In contrast to this, the tabu search heuristic performs good in the case of problem categories (b), (d) and (f) in which due to large setup costs lot building plays a key role for finding cheap production plans. If this is not the case and holding costs should be considered primarily, the tabu search heuristic does not satisfy.

Table 7: Deviation of the tabu search heuristic, $\nu = 1$

	(a)	(b)	(c)	(d)	(e)	(f)	average
L/E/1	0.00	0.00	0.00	0.00	0.00	0.00	0.00
L/A/1	10.34	1.50	7.89	0.79	5.88	0.76	4.53
A/E/1	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A/A/1	12.86	2.18	6.82	0.00	4.48	0.73	4.51
D/E/1	12.50	2.56	8.33	1.03	5.13	0.46	5.00
D/A/1	19.05	2.93	14.63	1.57	10.91	0.74	8.31
G/E/1	8.45	2.48	0.00	0.00	2.19	1.62	2.46
G/A/1	8.89	1.85	7.41	1.01	3.49	0.70	3.89
average	9.01	1.69	5.64	0.55	4.01	0.63	3.59

Table 8: Deviation of the tabu search heuristic, $\nu = 2$

	(a)	(b)	(c)	(d)	(e)	(f)	average
L/E/2	32.48	4.65	27.78	1.82	13.33	3.20	13.88
L/A/2	77.78	7.73	48.72	2.23	37.86	2.50	29.47
A/E/2	0.00	7.55	2.33	1.13	3.61	1.89	2.75
A/A/2	80.63	11.43	44.68	2.12	29.08	1.16	28.18
D/E/2	26.97	5.10	13.21	3.25	8.54	1.37	9.74
D/A/2	67.16	5.04	51.19	3.06	38.60	1.18	27.70
G/E/2	9.38	0.00	2.74	0.00	5.34	1.19	3.11
G/A/2	40.63	3.77	32.43	1.63	20.00	1.07	16.59
average	41.88	5.66	27.88	1.90	19.55	1.69	16.43

Table 9: Deviation of the tabu search heuristic, $\nu = 3$

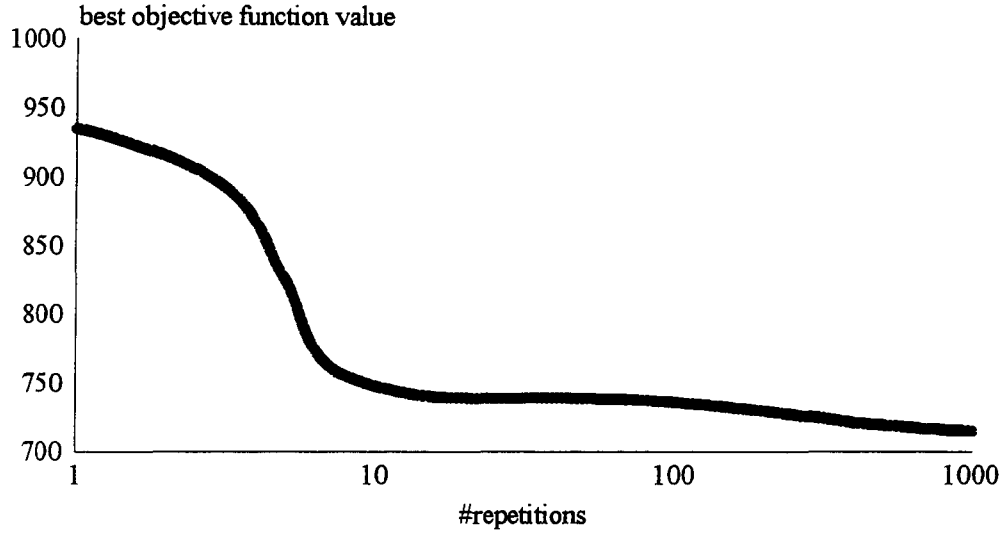
	(a)	(b)	(c)	(d)	(e)	(f)	average
L/E/3	26.80	6.00	23.21	2.50	22.47	4.63	14.27
L/A/3	71.11	6.09	60.95	3.86	47.52	3.53	32.18
A/E/3	11.01	3.31	9.52	0.83	0.78	0.00	4.24
A/A/3	50.85	5.49	44.44	2.09	28.16	1.51	22.09
D/E/3	39.07	7.18	32.14	5.02	24.80	2.82	18.51
D/A/3	72.82	10.24	57.50	5.72	46.39	1.55	32.37
G/E/3	0.45	0.00	0.00	0.00	1.96	1.49	0.65
G/A/3	38.10	3.14	33.13	2.15	20.55	1.38	16.41
average	38.78	5.18	32.61	2.77	24.08	2.11	17.59

The explanation for this phenomenon is obvious: While the randomized regret based heuristic does not allow idle periods during the construction phase as long as there is any unsatisfied demand, the tabu search heuristic enforces idle periods because of the underlying order in which demands are to be

considered. The former one thus tends to find narrow production plans which is a good thing to do when there is high workload while the latter one should be preferred if idle periods do not bear the risk of infeasibility.

Since both heuristics are complementary and fast it seems to be a good idea to employ both, that is, running the randomized regret based heuristic and the tabu search heuristic even if run-time is almost doubled. Observing the process of finding a feasible solution makes this approach attractive: In most cases, both heuristics find a production plan that is near to the final solution (after 1000 repetitions) quite soon. Figure 8 shows a typical decline of the objective function value's upper bound when these heuristics run.

Figure 8: Problem instance A/E/2 (a) when solved with the randomized regret based heuristic



So if both heuristics are combined, it would be possible to perform half of the number of repetitions to keep the run-time almost unchanged. Of course we pay a tribute for this, i.e. some instances may not be solved as good as before (or even no feasible solution might be found any more). In Tables 10 to 12 we provide the results when both heuristics are applied to a problem instance and the best solution is chosen afterwards. The number of repetitions is 500 for each heuristic so that the run-time equals the above measures. As it can be seen, the average deviation from the optimum objective function value is drastically reduced.

Table 10: Deviation of the combined heuristics, $\nu = 1$

	(a)	(b)	(c)	(d)	(e)	(f)	average
L/E/1	0.00	0.00	0.00	0.00	0.00	0.00	0.00
L/A/1	10.34	1.50	7.89	0.79	5.88	0.76	4.53
A/E/1	0.00	0.00	0.00	0.00	0.00	0.00	0.00
A/A/1	12.86	2.18	6.82	0.00	4.48	0.73	4.51
D/E/1	12.50	2.56	8.33	1.03	5.13	0.46	5.00
D/A/1	19.05	2.93	14.63	1.57	10.91	0.50	8.26
G/E/1	8.45	2.48	0.00	0.00	2.19	1.62	2.46
G/A/1	8.89	1.85	7.41	1.01	3.49	0.70	3.89
average	9.01	1.69	5.64	0.55	4.01	0.60	3.58

Table 11: Deviation of the combined heuristics, $\nu = 2$

	(a)	(b)	(c)	(d)	(e)	(f)	average
L/E/2	20.51	4.65	0.00	1.82	6.67	3.20	6.14
L/A/2	14.29	7.73	14.10	2.23	10.68	2.50	8.59
A/E/2	5.59	7.55	2.33	1.13	3.61	1.89	3.68
A/A/2	20.00	11.43	15.96	2.12	16.31	1.16	11.16
D/E/2	12.36	6.37	11.32	3.25	18.29	1.37	8.83
D/A/2	23.88	6.47	20.24	3.93	18.42	2.36	12.55
G/E/2	0.00	0.00	0.00	0.00	5.34	1.19	1.09
G/A/2	15.63	3.77	18.02	1.63	20.00	1.07	10.02
average	14.03	6.00	10.25	2.01	12.42	1.84	7.76

Table 12: Deviation of the combined heuristics, $\nu = 3$

	(a)	(b)	(c)	(d)	(e)	(f)	average
L/E/3	12.37	4.00	10.71	2.50	5.62	4.63	6.64
L/A/3	17.78	6.09	18.10	3.86	13.48	3.53	10.47
A/E/3	1.83	3.31	1.59	0.83	0.78	0.00	1.39
A/A/3	20.34	5.49	20.00	2.09	16.99	1.51	11.07
D/E/3	27.15	7.18	13.10	5.02	22.40	2.82	12.94
D/A/3	28.16	9.34	25.83	4.93	21.69	1.55	15.25
G/E/3	0.45	0.00	0.00	0.00	8.82	1.49	1.79
G/A/3	17.69	3.14	19.63	2.15	26.88	1.38	11.81
average	15.72	4.82	13.62	2.67	14.58	2.11	8.92

6 Conclusion and Future Work

In this paper we have presented a mixed-integer model and two heuristics for multi-level, single-machine lot sizing and scheduling problems - a randomized regret based heuristic and a tabu search heuristic. Both of them construct production plans in a backward oriented manner. The former one uses some kind of priority rule while the latter one uses a graph representation that guides the construction phase. A computational study brought out that both heuristics perform with equal run-time performance as well as with equal solution quality. Surprisingly, both heuristics are complementary, i.e. the randomized regret based heuristic performs good in those cases in which the tabu search gives poor results and vice versa. A combination of both drastically reduces the average deviation from the optimum objective function value. The combination gives the best results so far known. Furthermore, we discussed the importance of splitting lots to find good or even feasible production plans. Moreover, we pointed out that local search must not be restricted in feasible regions only and proved that tunneling is a necessary strategy in our application.

Future work shall refine the presented concepts. Other heuristics (e.g. simulated annealing) should be tested against the tabu search strategy. The problem itself is to be extended for meeting further requirements such as multiple resources, backorders or setup times.

Acknowledgement

The work on this paper was stimulated by Andreas Drexl who - as always - got very interested in discussing many details.

References

Andler, K., (1929), Rationalisierung der Fabrikation und optimale Losgröße, München, Oldenbourg

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

Bahl, H. C., Ritzman, L. P., Gupta, J. N. D., (1987), Determining Lot Sizes and Resource Requirements: A Review, Operations Research, Vol. 35, pp. 329-345

Balas, E., (1969), Machine Sequencing via Disjunctive Graphs: An Implicit Enumeration Algorithm, Operations Research, Vol. 17, pp. 941-957

Bollobás, B., (1979), Graph Theory – An Introductory Course, New York, Springer

Brüggemann, W., Jahnke, H., (1994), DLSP for 2-Stage Multi-Item Batch Production, International Journal of Production Research, Vol. 32, pp. 755-768

Dell'Amico, M., Trubian, M., (1993), Applying Tabu-Search to the Job-Shop Scheduling Problem, Annals of Operations Research, Vol. 41, pp. 231-252

Domschke, W., Scholl, A., Voß, S., (1993), Produktionsplanung – Ablauforganisatorische Aspekte, Heidelberg, Springer

Drexl, A., Haase, K., (1992), A New Type of Model for Multi-Item Capacitated Dynamic Lotsizing and Scheduling, Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 286

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

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

Faigle, U., Kern, W., (1992), Some Convergence Results for Probabilistic Tabu Search, ORSA Journal on Computing, Vol. 4, pp. 32-37

Glover, F., (1989), Tabu Search – Part I, ORSA Journal on Computing, Vol. 1, pp. 190-206

Glover, F., (1990), Tabu Search – Part II, ORSA Journal on Computing, Vol. 2, pp. 4-32

Glover, F., (1994), Tabu Search Fundamentals and Uses, Working Paper, University of Colorado

Haase, K., (1994), Lotsizing and Scheduling for Production Planning, Ph.D. thesis, University of Kiel, Lecture Notes in Economics and Mathematical Systems, Vol. 408, Berlin, Springer

Harris, F. W., (1913), How many parts to make at once, Factory, The Magazine of Management, Vol. 10, pp. 135-136, 152, reprinted (1990), Operations Research, Vol. 38, pp. 947-950

Helber, S., (1993), Kapazitätsorientierte Losgrößenplanung in PPS-Systemen, Ph.D. thesis, University of Munich

Hertz, A., de Werra, D., (1990), The Tabu Search Metaheuristic: How we used it, Annals of Mathematics and Artificial Intelligence, Vol. 1, pp. 111-121

Ivogun, P., Atkins, D., (1993), A Lower Bound and an Efficient Heuristic for Multistage Multiproduct Distribution Systems, Management Science, Vol. 39, pp. 204-217

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

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

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

Nowicki, E., Smutnicki, C., (1993), A Fast Taboo Search Algorithm for the Job Shop Problem, Working Paper, Instytut Cybernetyki Technicznej Politechniki Wroclawskiej

Roll, Y., Karni, R., (1991), Multi-Item, Multi-Level Lot Sizing with an Aggregate Capacity Constraint, European Journal of Operational Research, Vol. 51, pp. 73-87

Roundy, R. O., (1993), Efficient, Effective Lot Sizing for Multistage Production Systems, Operations Research, Vol. 41, pp. 371-385

Salomon, M., (1991), Deterministic Lot Sizing Models for Production Planning, Lecture Notes in Economics and Mathematical Systems, Vol. 355, Berlin, Springer

Salomon, M., Kuik, R., van Wassenhove, L.N., (1993), Statistical Search Methods for Lotsizing Problems, Annals of Operations Research, Vol. 41, pp. 453-468

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

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

Tempelmeier, H., Derstroff, M., (1993), Mehrstufige Mehrprodukt-Losgrößenplanung bei beschränkten Ressourcen und genereller Erzeugnisstruktur, OR Spektrum, Vol. 15, pp. 63-73

Appendix A: About the Necessity for Splitting Lots

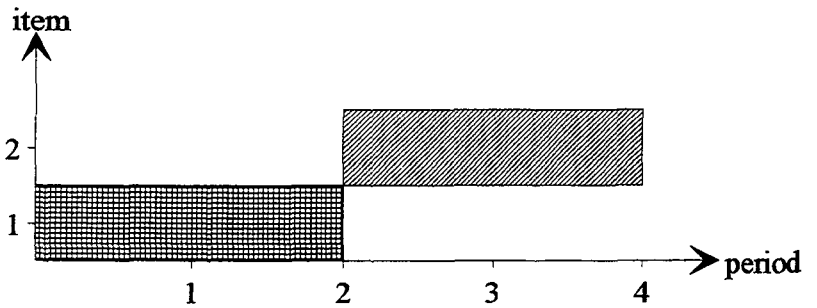
Suppose that we have two items to be produced on one machine which is initially setup for none of the two items. Let $T = 4$, $C_t = 20$ for all t and $p_j = 1$ for all j. Assume a single-level product structure. Table 13 provides the external demands as well as the setup and holding costs.

Table 13: Demands and costs of the example

	$t = 1$	$t = 2$	$t = 3$	$t = 4$	h_j	s_j
$j = 1$		20	20		5	90
$j = 2$				40	2	70

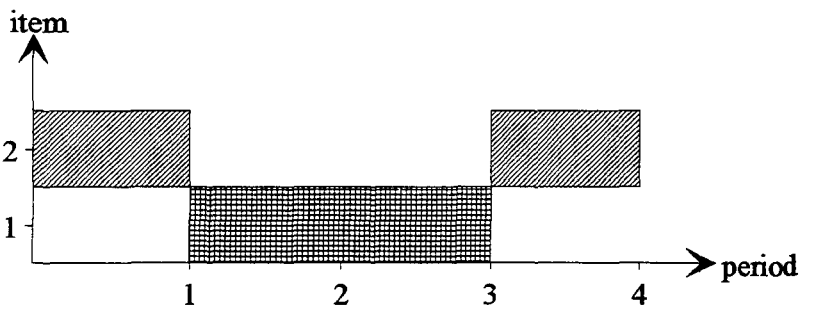
If we do not allow the splitting of lots we get the solution depicted in Figure 9 with an objective function value of 400.

Figure 9: A solution without lot splitting



If we do allow the splitting of lots the solution shown in Figure 10 with an objective function value of 350 would be possible.

Figure 10: A solution with lot splitting



Appendix B: About the Need for Tunneling

Suppose a single-level problem instance with three periods of time and two items. Assume 20 capacity units being available per period while the production of one item consumes one capacity unit. The machine shall not be setup for any item when we start. Let the demand be given as defined by Table 14.

Table 14: Demand matrix for the example

	t = 1	t = 2	t = 3
j = 1		10	10
j = 2		10	10

The order in which the four demand matrix entries are scheduled by our backward oriented scheme is defined by a tuple (o_{12} o_{13} o_{22} o_{23}) which is a permutation of {0, 1, 2, 3}: If o_{jt} equals zero, d_{jt} is scheduled first, if o_{jt} equals one, d_{jt} is scheduled second and so on. For instance, the tuple (3 0 2 1) that assigns $o_{12} = 3$, $o_{13} = 0$, $o_{22} = 2$ and $o_{23} = 1$ defines the production plan in Figure 11 (a).

In the following we use a matrix notation of the form

$$\begin{bmatrix} o_{12} & o_{13} \\ o_{22} & o_{23} \end{bmatrix}$$

to abbreviate the tuple notation. E.g. the matrices

$$\begin{bmatrix} 3 & 0 \\ 2 & 1 \end{bmatrix} \text{ and } \begin{bmatrix} 3 & 0 \\ 1 & 2 \end{bmatrix}$$

define the production plans pictured in Figure 11 where both plans were constructed by scheduling the demand for item 1 in period 3 at first (since o_{13} equals 0 in both cases) and by scheduling the demand for item 1 in period 2 at last (since o_{12} equals 3 in both cases). The plans differ in the order in which the demands for item 2 are scheduled.

Figure 11: The production plans that correspond to (a) $\begin{bmatrix} 3 & 0 \\ 2 & 1 \end{bmatrix}$ and to (b) $\begin{bmatrix} 3 & 0 \\ 1 & 2 \end{bmatrix}$, respectively

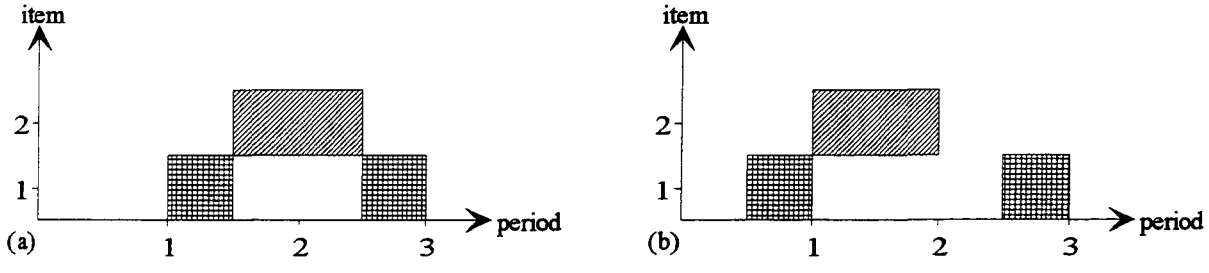


Figure 12 provides an overview of all 24 permutations where those matrices which define a feasible production plan are surrounded by solid lines and those which define an infeasible solution are surrounded by dotted lines (remember that at most one changeover can be done within each period). With respect to the graph rewriting scheme defined above the entries of a matrix define the total order of the root nodes (and indeed the matrix entries are nothing else than the reference count entries of the root nodes). The arcs in Figure 12 thus show which solution can be transformed into what solution by applying one transformation step, i.e. by redirecting exactly one arc. The solution $\begin{bmatrix} 3 & 0 \\ 2 & 1 \end{bmatrix}$ can for instance be transformed into the solution $\begin{bmatrix} 3 & 0 \\ 1 & 2 \end{bmatrix}$ in one step. Since we want to show that the set of feasible solutions is disconnected, the arcs show moves originating from feasible solutions only. If a move leads to an infeasible solution, the corresponding arc is dotted. Now it is easy to see, that starting in one feasible solution one cannot reach any other feasible solution bypassing feasible solutions only. For example, if we start with the solution $\begin{bmatrix} 1 & 0 \\ 3 & 2 \end{bmatrix}$ we can transform it into the solution $\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix}$ by performing basic transformations that pass no infeasible solution, but we cannot reach $\begin{bmatrix} 3 & 2 \\ 1 & 0 \end{bmatrix}$ without doing so. Hence it is proven, that tunneling is necessary in this application.

Figure 12: Feasible solutions are typically not connected