Drexl, Andreas; Haase, Knut

Sequential-analysis-based randomized-regret-methods for lotsizing and scheduling

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 323

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

Nr. 323

Sequential-Analysis-Based Randomized-Regret-Methods for Lotsizing and Scheduling

Drexl, A. and K. Haase

July 1993

Andreas Drexl, Knut Haase, Institut für Betriebswirtschaftslehre, Christian-Albrechts-Universität zu Kiel, Olshausenstr. 40, 2300 Kiel 1, F.R.G.

Abstract: Lotsizing and scheduling comprises activities which have to be done repeatedly within MRP-systems. We consider the proportional (multi-item, capacitated, dynamic) lotsizing and scheduling problem which is more general than the discrete lotsizing and scheduling problem as well as the continuous setup lotsizing problem. A simple stochastic backward algorithm with biased sampling via randomized regrets is presented. We partition the parameter space of the stochastic algorithm and choose subspaces via sequential analysis based on hypothesis testing. The new ideas provided in this paper, i.e. the randomized-regrets-based backward algorithm as well as the controlled search via sequential analysis, have three important properties: They are simple, effective, and rather general. Computational results are presented as well.

Keywords: Lotsizing and scheduling, backward algorithm / local search, biased sampling via randomized regrets / Monte Carlo simulation, hypothesis testing / sequential analysis

1. Introduction

Lotsizing and scheduling addresses the problem of determining the sequence and size of production lots (batches) for one or more items on one or more machines in a single- or multi-stage manufacturing system. The time horizon usually is divided into a finite number of time periods with equal length. There are capacity constraints for the machines. Demand is assumed to be known per period. The problem is to find a minimal cost production schedule, where machine setup and holding costs for inventory are considered to be relevant. Note that lotsizing and scheduling relates to short term production scheduling.

Most of the literature is dealing only with lotsizing, i.e. scheduling (sequencing) must be addressed separately. Especially the "Capacitated Lot-Sizing Problem" (CLSP) has been investigated intensively; see e.g. **Campbell, Mabert** 1991, **Chen, Thizy** 1990, **Diaby, Bahl, Karwan, Zionts** 1992, **Eppen, Martin** 1987, **Pochet, Wolsey** 1991 as well as **Thizy, Van Wassenhove** 1985.

The main difference between the Economic Lotsizing and Scheduling Problem (ELSP; see e.g. **Carreno** 1990, **Dobson** 1987, **Elmaghraby** 1978) not dealt with here and the models considered here is that the ELSP assumes the time-scale to be continuous, the demand for items to be constant and the planning horizon to be infinite.

The paper is organized as follows: In Section 2 we formulate the proportional, multi-item, capacitated, dynamic lotsizing and scheduling problem and discuss the relations to other lotsizing and scheduling models. Section 3 provides a simple backward algorithm using randomized-regret-based biased sampling. In Section 4 we discuss how to partition the and search within the parameter space of the method via hypothesis testing / sequential analysis. In Section 5 we relate the method presented here to other well-known local search methods. Section 6 provides computational results. We finish in Section 7 with a summary and an outline of future work.

2. Lotsizing and Scheduling Models

The models to be considered in this section are based on the following **general assumptions:** We consider a single-stage system, where a number of different items j = 1,...,J have to be manufactured on one machine (corresponding to a single capacity constraint). The time horizon T is segmented into a finite number of time periods t = 1,...,T with equal length. Demand d_{jt} for item j in period t is given. The setup costs s_j for item j incur whenever production of a batch starts. The holding costs h_j per unit of item j are identical for all periods.

The **Proportional Lotsizing and Scheduling Problem (PLSP)** is based on the following **specific assumption:**

A1. At most one changeover is allowed within each period.

According to assumption A1 we have to compute continuous lot sizes over one or several, adjacent or non-adjacent periods. The setup state can be preserved over idle periods. In addition setup costs are calculated by looking back several periods.

Mathematically the PLSP can be stated as follows:

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

$$\text{s.t.} \quad I_{j,t-1} + q_{jt} - I_{jt} = d_{jt} \qquad (j = 1,...,J;\ t = 1,...,T) \tag{2}$$

$$\sum_{j=1}^{J} v_{jt} \leq 1 \qquad (t = 1,...,T) \tag{3}$$

$$x_{jt} - v_{jt} + v_{j,t-1} \geq 0 \qquad (j = 1,...,J;\ t = 1,...,T) \tag{4}$$

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

$$\sum_{j=1}^{J} \pi_j q_{jt} \leq C_t \qquad (t = 1,...,T) \tag{6}$$

$$I_{jt}, q_{jt}, x_{jt} \geq 0 \qquad (j = 1,...,J;\ t = 1,...,T) \tag{7}$$

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

where

B : big number

C_t : the capacity (time) of the machine in period t,

d_{jt} : the demand for item j in period t,

h_j : the costs of keeping one unit of item j in inventory for one period

s_j : the setup costs for item j,

π_j : the time to produce one unit of item j,

I_{jt} : the inventory of item j at the end of period t,

q_{jt} : the quantity (number of units) of item j to be produced in period t,

x_{jt} : a variable indicating whether setup for item j takes place in period t ($x_{jt} > 0$) or not ($x_{jt} = 0$),

v_{jt} : a binary variable indicating the setup state.

(1) minimizes the sum of setup and holding costs. (2) corresponds to ordinary inventory balance constraints. (3) secures that the machine is setup for at most one item in each period. (4) couples the setup variables x_{jt} and the setup state variables v_{jt}. Since $s_j > 0$ we observe that x_{jt} takes only binary values. (5) couples the production decisions with the setup state of the machine. (6) secures feasibility with respect to the machine capacity. Note that π_j is set equal to one w.l.o.g., i.e. demand is given in capacity units.

In the PLSP we distinguish between two items to be producible in period t. The first item being produced in period t corresponds to the second item being produced in period t-1. For the second item the machine is going to be setup in period t after a number of time units *proportional* to the time used for producing the first item. The splitting of the machine capacity for the production of two products within one period proportional to the quantities needed motivates the name of the model. Noteworthy to say, that an equivalent formulation of the PLSP might use quantities q^1_{jt} (q^2_{jt}) for the first (the second) item; see **Drexl, Haase** 1992.

Note that the v_{j0} are given data (w.l.o.g. $v_{j0} := 0 \ \forall$ j) and the machine can remain in any status at the end of the horizon. In addition we assume that $I_{j0} := I_{j,T+1} := 0 \ \forall$ j.

The PLSP has been introduced in **Drexl, Haase** 1992, where several generalizations have been presented as well (the PLSP with setup times, with sequence-dependent setup costs, with multiple machines as well as the multi-stage PLSP).

Now we are going to briefly discuss two closely related lotsizing and scheduling models.

The **Discrete Lotsizing and Scheduling Problem (DLSP)** is based on the following **specific assumption:**

A2. The production process always runs full periods without changeover.

This "all or nothing production" implies that at most one item can be produced per period, setup costs have to be paid at the beginning of the periods only and lot sizes are multiples of a full period production. Like the PLSP the DLSP addresses short term production scheduling with periods as days or shifts.

The first mixed binary programming models based on assumption A2 have been developed some decades ago, but there was no method for solving medium- to large-size instances. Recently, an exact branch-and-bound approach based on Lagrangean

relaxation of the capacity constraints has been presented by **Fleischmann** 1990. The resulting subproblems have been solved by dynamic programming. This approach is capable of solving problems with (e.g.) J = 12, T = 122 or J = 3, T = 250 to optimality on a (slow) personal computer within a reasonable amount of time. This approach has been extended to the DLSP with sequence-dependent setup costs in **Fleischmann** 1992. Column generation based heuristics for the DLSP with setup times are presented in **Cattrysse, Salomon, Kuik, Van Wassenhove** 1990. A six field notation for the short classification of a variety of DLSPs is presented in **Salomon** 1991 as well as **Salomon, Kroon, Kuik, Van Wassenhove** 1991. Additionally, in both sources the complexity of feasibility problems associated with the DLSP are addressed. It is proved that the one machine DLSP with nonzero setup times as well as the parallel machine problem with zero setup times are NP-complete.

The **Continuous Setup Lotsizing Problem (CSLP)** is based on the following **specific assumption:**

A3. At most one item can be produced per period.

This implies that either there is one setup at the beginning of a period or no setup occurs at all within the period. Moreover, lotsizes are continuous quantities between zero and the maximum possible production of item j in period t. Like the PLSP and the DLSP, the CSLP relates to short term production scheduling.

In **Karmarkar, Schrage** 1985 a branch-and-bound approach based on Lagrangean relaxation of the capacity constraints has been presented, along with rather discouraging computational experiences even for small-sized problems. MPSARX, a mathematical programming system with automatic cut generation capabilities (**Pochet, Wolsey** 1991) allows to solve medium-sized problems to optimality.

Let us now briefly discuss the relations between the DLSP, the CSLP as well as the PLSP. Regarding the "all or nothing production" assumption of the DLSP and the feasibility of continuous lot sizes for the CSLP it follows that the set of feasible solutions X_{DLSP} of the DLSP is a subset of the set of feasible solutions X_{CSLP} of the CSLP. In addition it is easy to verify that the CSLP is a special case of the PLSP. Thus X_{CSLP} is a subset of the set of feasible solutions X_{PSLP} of the PLSP, which implies $X_{DLSP} \subseteq X_{CSLP} \subseteq X_{PLSP}$. These facts relating to sets of feasible solutions lead to the inequalities

$$Z^*_{DLSP} \geq Z^*_{CSLP} \geq Z^*_{PLSP}$$

related to the optimal objective function values Z^* of the different models.

Note that the different assumptions of the three models are not really relevant with respect to applications. Thus, despite the effort which has to be spent for the solution of the different models, the PLSP is considered to be superior to the DLSP and the CSLP regarding "quality" of solutions. Consequently, solving the PLSP fast (and "bad") may be even better than solving the DLSP or the CSLP to optimality.

3. Biased Sampling via Randomized Regrets

In the following we are going to describe a simple biased sampling method which essentially relies on randomized regrets. To start with, we calculate for each item j the cumulative demand (necessary capacity) D_{jt} from period t to the horizon T, which has to be satisfied in periods t, t-1, ..., 1:

$$D_{jt} := \max \{ 0, \sum_{\tau=t}^{T} (d_{j\tau} - q_{j\tau}) \} \quad (\forall\, j \text{ and } t)$$

In order to simplify the presentation we set $D_{j0} := 0\ \forall\, j$. The total demand (required capacity) which still has to be satisfied is specified by:

$$TD := \sum_{j=1}^{J} D_{j1}$$

For each item j the next period with positive demand in or before period t is:

$$nd_{jt} := \begin{cases} 0 & \text{if } t = 0 \\ t & \text{if } d_{jt} > 0 \quad (\forall\, j \text{ and } t) \\ nd_{j,t-1} & \text{otherwise} \end{cases}$$

The cumulative capacity from period $\tau = 1$ to $\tau = t$ is denoted by

$$\bar{C}_t := \sum_{\tau=1}^{t} C_\tau \quad (\forall\, t).$$

Scheduling will be done backward oriented. Moreover, if the machine is setup for item i in period t, item i will be scheduled until $D_{i,t-\tau} = 0$, where $\tau \geq 0$. That means, if the item i is scheduled over more than one period, say from period t to period $t-\tau$, then d_{it} as well as the demands $d_{i,t-1}, \ldots, d_{i,t-\tau}$ will be scheduled, too. In such a case $nd_{i,t-\tau-1}$ is the next period with positive demand for item i and we have to decide whether a machine state preserving "jump-back" from period $t-\tau$ to period $nd_{i,t-\tau-1}$ (target period) should occur in order to save setup costs or not.

A "jump-back" is not allowed, if it causes infeasibility indicated by $TD > \bar{C}_{nd_{i,t-1}}$. However, if a new item $j \neq i$ should be scheduled in period t we consider only those items with $D_{jt} > 0$. If we would change the setup state in period t to an item j with $D_{jt} = 0$, q_{jt} would become zero, too. In most cases, this would be a "nonoptimal" decision.

Scheduling an item j in period t and not in one of the preceeding periods $\tau = t, \ldots, 1$ saves at least $(h_j D_{jt})$ holding costs for item j. On the other hand, if the machine is not setup in period t for item j, setup costs of s_j are incurred. In such a case, the total "savings" are $(h_j D_{jt} - s_j)$ with respect to item j.

Thus, based on the setup state i of the machine in period t we define for all items j the following "convex" combination $r_{jt}(i)$ of the "savings" $(0 \leq \gamma \leq 1)$:

$$r_{jt}(i) := \begin{cases} (1-\gamma)\, h_j\, D_{jt} - \gamma \cdot s_j & \text{if } (j \neq i) \wedge (D_{jt} > 0) \quad \text{(i)} \\ (1-\gamma)\, h_j\, D_{j,nd_{j,t-1}} & \text{if } (j = i) \wedge (TD \leq \bar{C}_{nd_{j,t-1}}) \quad \text{(ii)} \\ -\infty & \text{otherwise} \quad \text{(iii)} \end{cases}$$

(i) occurs if the machine is not setup for the considered item and there exists in period t a cumulated demand which has to be satisfied in the periods t,t-1,...,1. (ii) occurs if the machine is setup for the considered item in period t and a "jump-back" to $nd_{j,t-1}$ is feasible. (iii) occurs for the item j in period t if $D_{jt} = 0$ and the machine is not setup for item j or a "jump-back" leads to infeasibility.

Finally we derive

$$\rho_{jt} := \begin{cases} 0 & \text{if } r_{jt}(i) = -\infty \\ (r_{jt}(i) - \min \{r_{kt}(i) \mid k = 1,...,J,\ r_{kt}(i) > -\infty\} + \epsilon)^{\delta} & \text{otherwise} \end{cases}$$

as decision criteria (with $\delta \geq 0$, $\epsilon > 0$). ρ_{jt} corresponds to the regret for producing item j not in period t or period $nd_{j,t-1}$, respectively. Only items, which are "feasible" get a positive regret and are "compared" with the worst feasible alternative (i.e. the item with smallest savings). The larger the regret of item j the more preferable it is to produce item j in period t. Note that $\epsilon > 0$ secures ρ_{jt} to be strictly positive while the exponential weight δ decreases or increases the differences between the ρ_{jt} for "small" or "large" δ, respectively.

The method for solving the PLSP which we are now going to present, incorporates the following three main ideas:

(a) Lotsizing and scheduling is done *backward* oriented starting in period T.
(b) In each period we *add* at most two items.
(c) The *regrets* ρ_{jt} are used in a *randomized* way in order to decide which item(s) to produce in a specific period.

A formal description of the stochastic backward add-method (BACKADD) may be given as follows (with explanations in braces):

BACKADD

Initialization:

$v_{jt} := 0$; ($\forall$ j and t = 0,1,...,T)
$q_{jt} := 0$; ($\forall$ j and t)
t := T; {start at the planning horizon}
i := 0; {the machine is not setup for any item}

while TD > 0 **and** t > 0 **do** {*start of loop 1*}

begin

determine ρ_{jt} ($\forall$ j);

if $\sum_{j=1}^{J} \rho_{jt} = 0$ **then** t := t - 1 {step backward if no scheduling is possible in t}

else begin {scheduling}

choose j* at random proportional to ρ_{jt};

i := j*; {the setup state changes to item j*}

if $y_{it} = 1$ **then** {check whether jump-back to "target-period" is required}

begin

$y_{i,t-1} := \ldots := y_{i,nd_{i,t-1}+1} := 1$; {fix setup state between t and target-period}

$t := nd_{i,t-1}$;

end;

while $D_{it} > 0$ **do** {*start of loop 2*}

begin

if $\sum_{j=1}^{J} y_{jt} = 0$ **then** $y_{it} := 1$ {determine setup for second item in period t}

else $y_{i,t-1} := 1$; {determine setup for first item in period t}

$q_{it} := \min \{C_t , D_{it}\}$; {determine production quantity}

$C_t := C_t - q_{it}$; {update capacity}

TD := TD - q_{it}; {update total demand which has to be satisfied}

update $D_{i\tau}$ ($\forall \tau = 1,\ldots,t$);

if $C_t = 0$ **then** t := t - 1; {step backward if no more capacity is available}

end; {*end of loop 2*}

if $\sum_{j=1}^{J} y_{j,t-1} = 1$ **then** set t := t - 1 {step backward if two items are scheduled}

end; {*end of loop 1*}

if TD > 0 **then** Z := ∞, else evaluate feasible solution.

* * * * *

Note that BACKADD determines *one* solution (corresponding to an upper bound) of the PLSP per execution. #exc (repeated) executions of BACKADD in general yield several ($\leq$ #exc) different solutions.

Recently, Monte Carlo type sampling methods for solving 0-1 programming problems have attracted increasing attention. In **Drexl** 1991 an efficient (randomized-regret-based, too) Monte Carlo method for solving assignment type project scheduling problems has been presented. **Bertocchi, Brandolini, Slominski, Sobczynska** 1992 introduce a two-phase Monte Carlo approach (based on non-asymptotic order statistics) for solving general 0-1

programming problems. Note that the PLSP, a mixed-integer programming problem, seems to be more challenging to sampling methods than pure discrete optimization problems due to the existence of continuous as well as integer variables.

It should be noted here that backward lotsizing and scheduling methods are superior to forward-oriented ones (like the famous **Dixon, Silver** 1981 heuristic as an example) in the capacitated case from a conceptual point of view: There is no need for complicated and time-consuming look ahead procedures in order to secure resource feasibility.

As will be seen later on, the performance of BACKADD depends on the choice of the parameters γ and δ, i.e. on the weight of the setup / holding costs part as well as on the randomization scheme.

In order to support the choice of the parameters we suppose to partition the parameter space as follows:

Let γ and δ be real-valued (≥ 0) parameters with partition

$[\gamma_0, \gamma_1), [\gamma_1, \gamma_2), \ldots, [\gamma_i, \gamma_{i+1}), \ldots, [\gamma_{I-1}, \gamma_I)$ and

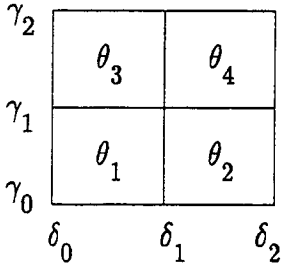
$[\delta_0, \delta_1), [\delta_1, \delta_2), \ldots, [\delta_k, \delta_{k+1}), \ldots, [\delta_{K-1}, \delta_K)$.

Then we construct the following cross products (see the instance partitioning provided in Figure 1 for I = K = 2):

$$\theta_1 = [\gamma_0, \gamma_1) \times [\delta_0, \delta_1)$$
$$\theta_2 = [\gamma_0, \gamma_1) \times [\delta_1, \delta_2)$$
$$\vdots$$
$$\theta_h = [\gamma_i, \gamma_{i+1}) \times [\delta_k, \delta_{k+1})$$
$$\vdots$$
$$\theta_H = [\gamma_{I-1}, \gamma_I) \times [\delta_{K-1}, \delta_K)$$

FIGURE 1:

Partition of the parameter space for I = K = 2



Thus $\{\theta_1, \ldots, \theta_H\}$ corresponds to a partition of the parameter space Θ such that $\Theta := \bigcup_{h=1}^{H} \theta_h$ and $\theta_i \cap \theta_j = \phi$ for $i \neq j$. Note that spaces of higher dimensions may be partitioned in the same way as well.

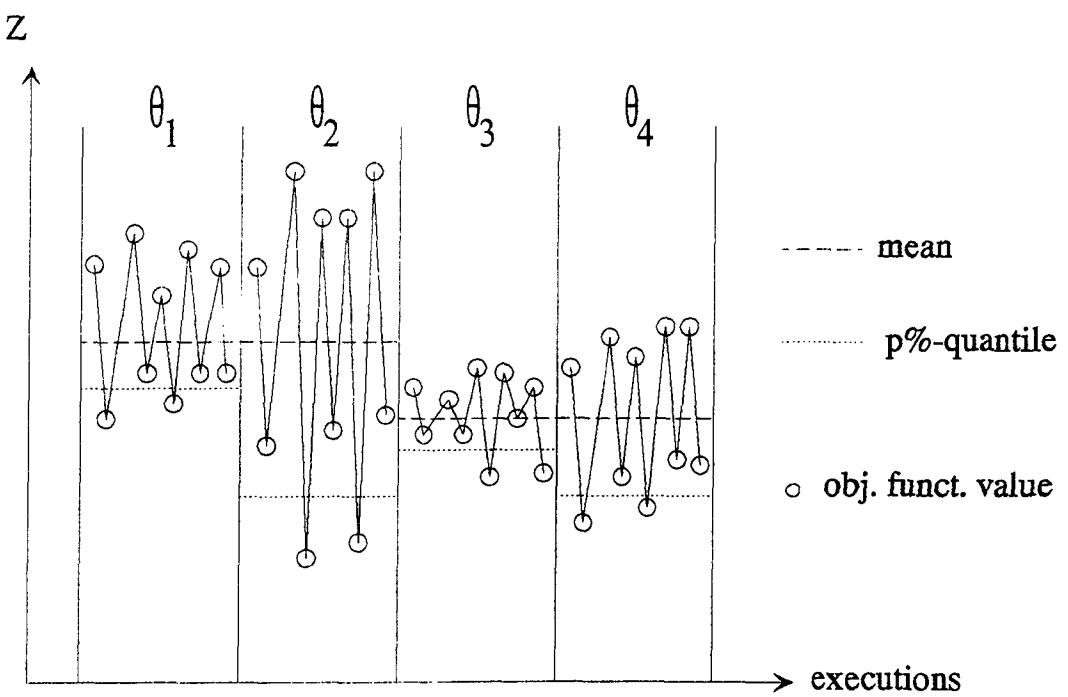
In the following we need some additional symbols: x denotes a (feasible) solution of the PLSP, $Z(x)$ the objective function value associated with x.

The stochastic algorithm BACKADD(Θ), operating on the full parameter space Θ, tailors to BACKADD(θ_h) in the case where it only operates on the subspace θ_h. In addition the application of BACKADD(θ_h) reveals $Z(x(\theta_h))$ to become a random variable and $Z_1(x(\theta_h))$, $Z_2(x(\theta_h))$, ... to be a stochastic process.

Now the *fundamental question* arises, how (often) to choose θ_h , h = 1,...,H. Figure 2 provides a fictitious example demonstrating potential relationships between the mean and the p%-quantile associated with the random variables $Z(x(\theta_h))$, h = 1,...,4. It seems to be meaningless to take (an estimate of) the mean as the only criterion. Rather, an estimate of a combination of the mean and the standard deviation could be a good suggestion for choosing subspaces. Moreover, as Figure 2 indicates, as a single criterion the p%-quantile seems to be a good indicator for choosing subspaces; i.e. in subspaces θ_h for which the p%-quantile is relatively high, the probability of determining low cost objective function values is relatively small. Thus in the following we try to identify those subspaces θ_h of Θ which have a small p%-quantile. Therefore we will show how to use the theory and techniques of sequential analysis in order to control the choice of the subspaces θ_h of Θ.

FIGURE 2:

Objective function values depending on subspaces θ_h



4. Controlled Search via Sequential Analysis

Let Y be a Bernoulli distributed (discrete) random variable, i.e.

$$f_Y(y; p) := p^y (1-p)^{1-y} \quad (0 \le p \le 1)$$

with p being a probability. In addition let $Y_1, Y_2, ..., Y_n$ represent n identical independent Bernoulli distributed random variables. Denoting with y_i a realization of Y_i, the random variable

$$S_n := \sum_{i=1}^{n} y_i$$

is binomially distributed, denoted as $f_{S_n}(s_n; n, p)$, with density function:

$$f_{\mathfrak{X}}(r; n, p) := \begin{cases} \left[\begin{matrix} n \\ r \end{matrix} \right] \cdot p^r \cdot (1-p)^{n-r}, & \text{for } r = 0,1,...,n \\ 0, & \text{otherwise} \end{cases}$$

Moreover, let $f_Z(z)$ denote the density function of a continuous random variable Z, ξ_p the p%-quantile of $f_Z(z)$ and $z_1, z_2, ..., z_n$ a sample of $f_Z(z)$. Then the transformation

$$y_i := \begin{cases} 1, & \text{if } z_i \leq \xi_p \\ 0, & \text{otherwise} \end{cases} \quad \text{for } i = 1, 2, ..., n$$

yields $S_n = \sum_{i=1}^{n} y_i$ with density function $f_{S_n}(s_n; n, p)$.

Suppose that it is desired to test whether ξ_{p*} is less than or equal a specified value, say ξ. This leads to

$$\mathcal{H}_0: \xi_{p*} \leq \xi \text{ versus } \mathcal{H}_1: \xi_{p*} > \xi$$

where $\mathcal{H}_0$ is called the null-hypothesis, being tested, and $\mathcal{H}_1$ the alternative hypothesis. This test, concerning the quantile $f_Z(z)$, may be transformed into the following *hypothesis testing problem*

$$\mathcal{H}_0: p \geq p^* \text{ versus } \mathcal{H}_1: p < p^*,$$

concerning the probability $f_Y(y; p)$ with $0 \leq p^* \leq 1$. We

accept $\mathcal{H}_0$, if S_n/n is "large" or

reject $\mathcal{H}_0$, if S_n/n is "small"

where "large" and "small" obviously depend on p^*.

From sequential analysis several tests are known with which hypotheses testing may be performed; cf. e.g. **Siegmund** 1985, **Irle** 1990. Among them are curtailed tests, repeated significance tests as well as sequential probability ratio tests. In **Haase** 1993 these tests have been compared via simulation with respect to their capability of approximating the power function (probability of rejecting $\mathcal{H}_0$, expected number of repetitions of the test; both as a function of p^*). As a result the **sequential probability ratio test (SPRT)** seems to be most suited.

The SPRT, which is based on the likelihood ratio, originally has been designed for simple hypotheses only. Thus for our composite hypothesis we have to make the following transformations:

$\mathcal{H}_0$: $p \geq p^*$ $\Rightarrow$ $\mathcal{H}_0$: $p = p^0$

$\mathcal{H}_1$: $p < p^*$ $\Rightarrow$ $\mathcal{H}_1$: $p = p^1$ with $p^1 < p^* \leq p^0$

Using the conventional symbols, i.e. α denotes the type I error (significance level) and β the type II error, we get the following stopping and decison rules for the SPRT:

Stopping rule: Stop at $\mathfrak{I} = \min \{ i \mid \mathfrak{h}_i \notin (A, B)\}$, where

$$\mathfrak{h}_i := \left[\frac{p^1}{p^0}\right]^{S_i} \left[\frac{1-p^0}{1-p^1}\right]^{S_i - i},$$

$$A := \frac{1-\alpha}{\beta} \text{ and } B := \frac{\alpha}{1-\beta}.$$

Decision rule: Reject $\mathcal{H}_0$, if $\mathfrak{h}_{\mathfrak{I}} \geq B$; accept $\mathcal{H}_0$, if $\mathfrak{h}_{\mathfrak{I}} \leq A$.

For the parameters α, β, p^0 as well as p^1 we set in the following $\alpha = 0.05$, $\beta = 0.1$, $p^0 = 0.2$ as well as $p^1 = 0.05$, respectively.

In the following BACKADD/SEQ denotes the method BACKADD where the choice of subspaces θ_h is controlled via sequential analysis, i.e. the SPRT.

5. Relations to Local Search Methods

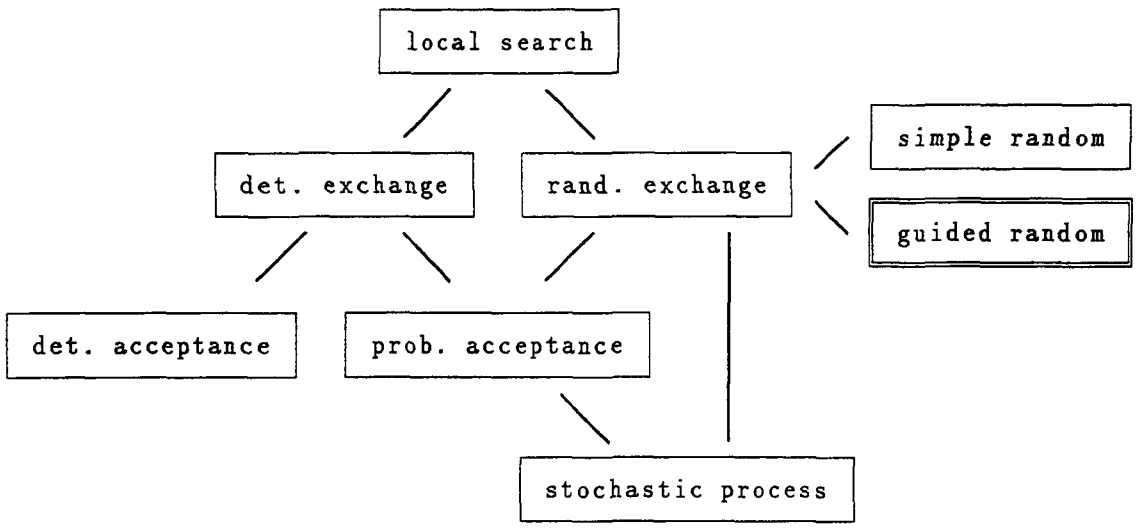
Local search provides extremely powerful methods for solving hard (combinatorial) optimization problems to suboptimality (for the complexity of local search see **Johnson, Papadimitriou, Yannakakis** 1988 as well as **Yannakakis** 1990). In the following we will relate the presented methods especially to simulated annealing, tabu search as well as to genetic algorithms.

In general, local search procedures consist of an exchange and an acceptance component (viz. generation and acceptance of moves). Starting from a current solution, the *exchange (or generation) component* provides neighbouring solutions. The *acceptance component* decides whether the new solution(s) should be accepted or not. From a conceptual point of view an exchange (or interchange) method consists out of a sequence of add and/or drop operations. Omitting one of the two reveals add and drop to be special cases of local search. Thus BACKADD essentially is a local search method.

In Figure 3 we first distinguish local search methods to be either deterministic or randomized exchange procedures. Second, both types of exchange methods may be either combined with a deterministic or with a probabilistic acceptance method. Exchange and/or acceptance probabilities may either be modified or used without modification. Randomized exchange and/or acceptance produces a sequence of objective function values corresponding to a stochastic process. Note that, despite tabu search, deterministic exchange in combination with deterministic acceptance is likely to be trapped in local optima. With respect to threshold accepting, in Figure 3 an additional arc between "rand. exchange" and "det. acceptance" could be introduced (cf. **Dueck, Scheurer** 1990).

Randomized exchange procedures may either be of a *simple or guided random* type. While in the former case each neighbour has the same probability of being generated, in the latter neighbours are generated with a probability controlled by some method. Thus BACKADD in conjunction with sequential analysis (for controlling parameter subspaces) corresponds to guided random exchange methods. (Up to our knowledge the term "guided random search" has been introduced by **Hartmann, van Hee** 1992 in the context of Markov decision process-based search.)

FIGURE 3: Local Search Methods



In **simulated annealing** (see, e.g. **Johnson, Aragon, McGeoch, Schevon** 1989, 1991, **van Laarhoven, Aarts, Lenstra** 1992, **Martin, Otto, Felten** 1992), the sequence of solutions does not converge monotonically towards a local optimum. Solutions randomly walk up and down with respect to the objective function value through the search space. Starting from a current solution, first a neighbouring solution is selected, either deterministically or at random. Then, based on the induced change of the objective function value, this transition is either accepted or rejected with a probability being controlled by some parameters. Clearly, the selection of neighbouring solutions may be based on randomized regrets (related to the respective changes of the objective function values) by making properly use of the sequential analysis motivated controlled search in the parameter space as described above.

In **tabu search** (see, e.g. **Dammeyer, Voß** 1992, **Dell'Amico, Trubian** 1992, **Glover** 1989, 1990, **Hertz, de Werra** 1990) one of the main ideas is to guide the search process deterministically out of local optima. This can be achieved if the best neighbouring solution will be accepted as a desirable move even if it is worsening the current solution. In order to prevent from cycling one of the essential ingredients of tabu search is to maintain a *tabu list* (of forbidden transitions between neighbouring solutions) throughout the search. Thus, basically tabu search can be considered as a deterministic exchange as well as a deterministic acceptance method. In addition, randomized exchange and/or

acceptance may be incorporated into tabu search as well, see e.g. **Faigle, Kern** 1992 as a recent reference to probabilistic tabu search. To be more specific, the selection of neighbouring solutions can also be based on randomized regrets - thus keeping track of the ideas presented in the paper for tabu search as well.

Genetic algorithms (see, e.g. **Dorndorf, Pesch** 1992, **Goldberg** 1989, **Liepins, Hilliard** 1989, **Mühlenbein, Gorges-Schleuter, Krämer** 1988 as well as **Pesch** 1993) are motivated by the analogy of evolution. They have been designed as general search methods working on populations of feasible solutions in order to make use of properties which "good" solutions have in common. Genetic algorithms mainly consist of three basic operators, namely reproduction, crossover and mutation employed to construct a new population. For *reproduction* an individual (coded as a string) is copied with probability proportional to its fitness value. Then *crossover* is applied to randomly partitioned pairs of the new population while (in a simple case) both strings are randomly cut at a certain position and tails are exchanged. Random changes to single elements of an individual, i.e. *mutation,* are of minor importance within genetic algorithms. Thus genetic algorithms may be seen to be randomized exchange methods, where the exchange process is guided via fitness values, in combination with deterministic or probabilistic acceptance. Clearly, fitness based reproduction of individuals may be replaced by taking randomized fitness regrets, which is closely related to scaling and windowing. In addition, the selection of pairs of the new population once more may be based on randomized fitness regrets. Last but not least, the choice of the position of the string may be based on randomized regrets. Thus genetic algorithms bear a wide range of potential applications of the ideas presented in this paper.

Note that our concept of guided random exchange is closely related to the "modification of generation probabilities" (**Faigle, Kern** 1992) as follows: Let us assume that the parameter space Θ defines in advance the whole set of probabilities for exchanging neighbouring solutions. Then, ceteris paribus, w.r.t. the subspace θ_h exchange takes place with some given generation probabilities (defined by θ_h). Changing from subspace θ_h to subspace θ_k conceptually is a method of "modification of generation probabilities". Thus the controlled search within the parameter space via sequential analysis as described in section 4 essentially is a general method of modifying exchange (or generation) probabilities.

6. Computational Results

First we are going to present an instance generator for the PLSP. A parametric characterization of the instances investigated below may be given as follows (for more details see **Haase** 1993):

J the number of items.

T^m the number of macro-periods.

T^s the number of sub-periods of each macro-period.

hc maximal holding costs coefficient, i.e. h_j is an integer drawn at random out of {1, 2, ..., hc}.

TBO time between order (positive integer).

ΔTBO a positive integer for the determination of the setup costs.

dp a symbol denoting the demand pattern, with dp = (η^1,η^2) being a tuple.

One demand (positive integer) can occur for each item within each macro-period. If η^1 equals

ud a demand will be generated at random with uniform distribution.

pt the pattern of the randomly generated demand has a positive trend.

ds a demand will be generated at random with uniform distribution whereby the expected values of the demand of the items are of different seize.

The second symbol denotes, in which sub-periods the demand occurs. If η^2 equals

e the demand occurs at the end (i.e. last sub-period) of the macro-periods only.

i the demand occurs at the end of a macro-period or T^s-1 sub-periods later only (at random with equal probability); in the last macro-period the demand always occurs in the last sub-period.

cp an integer for the capacity pattern. If cp equals

co we have a constant pattern over all periods.

ic the capacity is increasing.

U capacity utilization factor ($0 \leq U \leq 1$).

Finally n denotes the number of instances generated at random for a specific parametric characterization {J, T^m, T^s, U, h^m, TBO, dev, td, vd, tC}, hence INST(J,T^m, T^s,hc,TBO,ΔTBO,dp,cp,U;n) specifies a sample of instances under consideration.

Note that the distinction between macro-periods and sub-periods is necessary in order to ensure feasibility. If e.g. the number of sub-periods is less than the number of items with positive demand in the first macro-period, no solution exists because the number of different items producible is less than the number of items which must be produced in the first macro-period. In total the number of periods under consideration T equals $T^m \cdot T^s$.

Let $\bar{d}_j = 1/T \sum_{\tau=1}^{T} d_{j\tau}$, i.e. the average demand of item j. TBO and ΔTBO define a set A := {TBO-ΔTBO, ..., TBO+ΔTBO }. TBO_j is the approximate time between orders, drawn at random from A with uniform distribution. Thus the holding costs of a lot size with a length of TBO_j periods can be approximated by $[\bar{d}_j\, h_j \sum_{\tau=1}^{TBO_j} (\tau\text{-}1)]$. Now, let us assume that the TBO_j is as large that the approximate holding costs are equal to the setup costs. Thus the setup costs of item j are defined as follows:

$s_j := \lceil \bar{d}_j\, h_j\, TBO_j\, (TBO_j - 1) / 2 \rceil$ (j = 1,...,J) with $\lceil \alpha \rceil$ being the greatest integer smaller than α.

In the following we investigate the performance of BACKADD by a computational study. BACKADD has been implemented in Turbo Pascal 5.0 on a PS/2 Model 70 486. Since no tailor-made optimal solution approach for the PLSP exists we compute optimal benchmark solutions with the standard mixed-integer programming (MIP) solver LINDO. Only small instances of the PLSP with up to 50 binary variables are solvable in reasonable time with LINDO ($\leq$ two hours on a PS/2 Model 70 486).

Another inquiry is on the effect of the integration of the sequential test based parameter control. In BACKADD/SEQ we use the following partitioning: $\gamma_0 = 0$, $\gamma_1 = .33$, $\gamma_2 = .66$, $\gamma_3 = 1$ and $\delta_0 = 0$, $\delta_1 = 3$, $\delta_2 = 6$, $\delta_3 = 9$; i.e. Θ is partitioned into 9 subspaces $\theta_1, \ldots, \theta_9$.

For 16 parameterizations of INST(J,T^m,T^s,hc,TBO,ΔTBO,dp,cp,U;5) we generate n=5 instances each (cf. Table 1). We start with the basic parameterization INST(3,5,3,1,8,0,(ud,e),co,.8;5) (cf. S001 to S005 in Table 1). Then we consider a larger and a smaller capacity utilization, i.e. U=.95 and U=.65 whereby the other parameters are kept as in the basic parameterization (cf. S011 to S015 and S021 to S025 in Table 1). The instances S031 to S105 also differ in only one parameter position from the basic parameterization. In S111 to S115 the number of sub-periods per macro-period is increased from 3 to 4 and the number of macro-periods is decreased from 5 to 4. Thus the problems contain less than 50 binary variables (=J·T) which is necessary to determine an optimal PLSP solution within reasonable time. In S121 to S125 the number of items is decreased from 3 to 2 and the number of macro-periods is increased from 5 to 12. In S131 to S155 more than one parameters of the basic parameterization are changed. Such problems are supposed to be more complex than the preceding ones.

Each (small) problem (cf. Table 1) has been solved by BACKADD and BACKADD/SEQ using #exc=1000 executions each. Table 2 provides the average deviation between the optimal objective function value and the objective function value of the best solution of BACKADD and BACKADD/SEQ, denoted as $\bar{Z}$ and $\bar{Z}$/SEQ, respectively (average of 5 instances per entry). Furthermore the average fraction of infeasible solutions, denoted as INF and INF/SEQ, is given in Table 2, respectively.

In all 16 parameterizations with 5 instances the average deviation from optimum is less than one percent for BACKADD/SEQ. The objective function value of the solution determined with BACKADD (BACKADD/SEQ) for the problem S105 deviates from the optimal objective functions value by 11.9 (3.6) percent. Such a poor result occurred only once in our test. Thus the solution quality for small instances is very high. In the total average (all 16·5 instances) $\bar{Z}$ is .12% larger than $\bar{Z}$/SEQ, i.e. the integration of parameter control has improved (slightly) the solution quality. Moreover, the fraction of infeasible solutions is substantially decreased due to the integration of parameter control (cf. Table 2).

TABLE 1: Small instances

	J	T^m	T^s	hc	TBO	ΔTBO	dp	cp	U	n
S001 .. S005	3	5	3	1	8	0	(ud,e)	co	.8	5
S011 .. S015	3	5	3	1	8	0	(ud,e)	co	.95	5
S021 .. S025	3	5	3	1	8	0	(ud,e)	co	.65	5
S031 .. S035	3	5	3	1	8	0	(ud,e)	pt	.8	5
S041 .. S045	3	5	3	1	8	0	(ud,i)	co	.8	5
S051 .. S055	3	5	3	1	8	0	(pt,e)	co	.8	5
S061 .. S065	3	5	3	1	8	0	(ds,e)	co	.8	5
S071 .. S075	3	5	3	1	8	5	(ud,e)	co	.8	5
S081 .. S085	3	5	3	1	12	0	(ud,e)	co	.8	5
S091 .. S095	3	5	3	1	4	0	(ud,e)	co	.8	5
S101 .. S105	3	5	3	2	8	0	(ud,e)	co	.8	5
S111 .. S115	3	4	4	1	8	0	(ud,e)	co	.8	5
S121 .. S125	2	12	2	1	12	0	(ud,e)	co	.8	5
S131 .. S135	4	3	4	1	8	0	(ud,e)	co	.8	5
S141 .. S145	3	5	3	2	8	4	(ds,i)	pt	.8	5
S151 .. S155	3	4	4	3	8	4	(pt,i)	co	.8	5

TABLE 2: Results for small instances

	$\bar{Z}$	$\bar{Z}$/SEQ	INF	INF/SEQ
S001 .. S005	0	0	18	14
S011 .. S015	.76	.76	50	41
S021 .. S025	0	0	12	10
S031 .. S035	.25	.25	29	22
S041 .. S045	0	.06	8	1
S051 .. S055	.73	.05	19	9
S061 .. S065	0	0	13	9
S071 .. S075	0	0	10	8
S081 .. S085	0	0	46	34
S091 .. S095	.22	.98	12	8
S101 .. S105	2.4	.75	28	23
S111 .. S115	0	0	9	5
S121 .. S125	.08	.08	0	0
S131 .. S135	0	.03	24	21
S141 .. S145	1.06	0.06	1	2
S151 .. S155	.01	.01	1	1

TABLE 3: Large instances

	J	T^m	T^s	hc	TBO	ΔTBO	dp	cp	U	n
L000 .. L009	6	10	8	1	30	0	(ud,e)	co	.8	10
L010 .. L019	6	10	8	1	30	0	(ud,e)	co	.95	10
L020 .. L029	6	10	8	1	30	0	(ud,e)	co	.65	10
L030 .. L039	6	10	8	1	30	0	(ud,e)	ic	.8	10
L040 .. L049	6	10	8	1	30	0	(ud,i)	co	.8	10
L050 .. L059	6	10	8	1	30	0	(pt,e)	co	.8	10
L060 .. L069	6	10	8	1	30	0	(ds,e)	co	.8	10
L070 .. L079	6	10	8	1	30	15	(ud,e)	co	.8	10
L080 .. L089	6	10	8	1	60	0	(ud,e)	co	.8	10
L090 .. L099	6	10	8	1	15	0	(ud,e)	co	.8	10
L100 .. L109	6	10	8	4	30	0	(ud,e)	co	.8	10
L110 .. L119	6	10	12	1	30	0	(ud,e)	co	.8	10
L120 .. L129	6	10	6	1	30	0	(ud,e)	co	.8	10
L130 .. L139	6	5	8	1	30	0	(ud,e)	co	.8	10
L140 .. L149	6	15	8	1	30	0	(ud,e)	co	.8	10
L150 .. L159	3	10	8	2	30	0	(ud,e)	co	.8	10
L160 .. L169	10	10	12	1	30	0	(ud,e)	co	.8	10
L170 .. L179	4	25	5	3	50	20	(pt,i)	co	.9	10
L180 .. L189	9	10	11	6	50	20	(ds,i)	pt	.75	10
L190 .. L199	6	10	8	1	30	0	(ds,i)	co	.8	10

In order to analyze the effect of the sequential test we have generated 20·10 large lotsizing and scheduling problems with specific criteria as provided in Table 3. Again we start with a basic parameterization, i.e. INST(6,10,8,1,30,0,(ud,e),co,.8;10). Then we change successively the capacity utilization, the capacity pattern, and so on. For the problems L170 to L199 more than one parameter differs from the basic parameterization.

Each instance of Table 3 has been solved with BACKADD and BACKADD/SEQ. The average percentage deviations (10 instances per entry), denoted as $\Delta\bar{Z}$, of the best objective function values computed by BACKADD and BACKADD/SEQ are provided in Table 4. In addition, Table 4 contains the average number of infeasible solutions of BACKADD and BACKADD/SEQ, respectively.

Independently of the chosen parameterization the integration of parameter control has increased the solution quality (on the average). The improvement is not large. This may be reasoned by the efficiency of BACKADD: The results for the small instances (cf. Table 2) indicate that the best objective function value of BACKADD is near the optimal solution. It is thus difficult to determine a less costly solution. Whereas a heuristic which generates a solution with an objective function value which is much

larger than the optimal objective function value may easily be improved. Thus BACKADD seems to be efficient for large problems as well.

TABLE 4: Results for large instances

	$\Delta\bar{Z}$	INF	INF/SEQ
L000 .. L009	1.16	24.6	16.3
L010 .. L019	.67	33.2	26
L020 .. L029	.94	44.3	14.6
L030 .. L039	1.81	31.7	14.7
L040 .. L049	.34	10.3	5.7
L050 .. L059	1.61	46.1	31.3
L060 .. L069	.86	15.4	13.5
L070 .. L079	.51	18.8	13.7
L080 .. L089	1.07	17.9	16.8
L090 .. L099	1.14	12.4	4.8
L100 .. L109	1.04	21.4	17
L110 .. L119	1.27	13.1	7.1
L120 .. L129	.88	21.4	16.7
L130 .. L139	.98	3.9	3.9
L140 .. L149	.78	16.6	10
L150 .. L159	.72	0	0
L160 .. L169	.55	44.1	26
L170 .. L179	.12	41	39.8
L180 .. L189	.76	1	.9
L190 .. L199	1.02	10.6	9.3

Let us consider the problem L000 in more detail. In Figure 3 the sorted objective function values Z of 1000 executions of BACKADD and BACKADD/SEQ are plotted.

Figure 3 indicates that the number of infeasible solutions computed by BACKADD as well as by BACKADD/SEQ is about 100. BACKADD/SEQ has found some solutions the objective function values of which are substantially larger (smaller) than the largest (smallest) objective function value computed by BACKADD. In most cases, the objective function values determined by BACKADD/SEQ are smaller than the objective values of BACKADD. Thus the quality of solutions is increased due to the integration of the sequential test. Such a result was observable in most of the analyzed instances.

In Figure 4 the smallest objective function values Z(#exc) of the best solutions computed until to the #exc-th execution (#exc=1,...,1000) of BACKADD and BACKADD/SEQ are depicted.

FIGURE 3:
Sorted objective function values Z computed by BACKADD and BACKADD/SEQ for instance L000

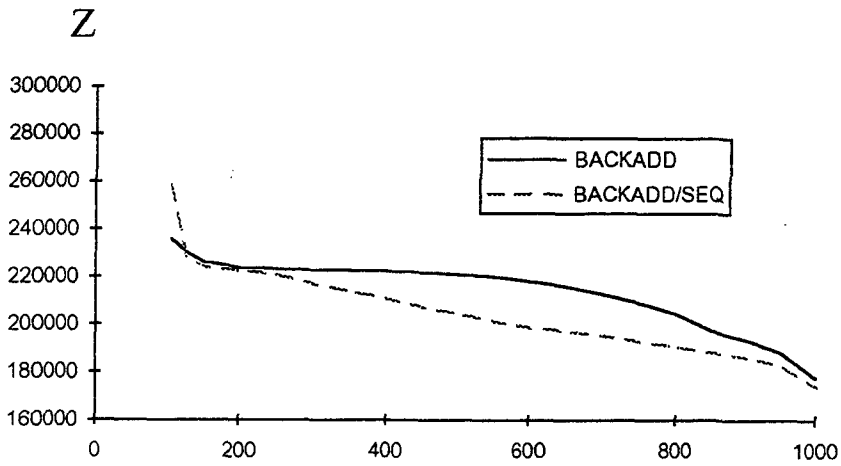
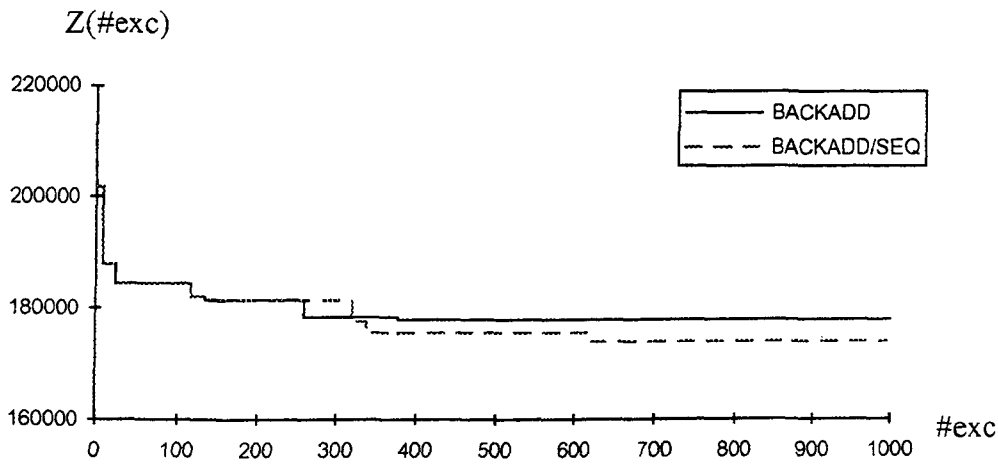


FIGURE 4:
Objective function values Z(# exc) of BACKADD and BACKADD/SEQ for instance L000



Recall that we have considered a partition of the parameter space into 9 subspaces. The expected sample size to reject or accept the null-hypothesis is around 30. So after $9 \cdot 30 = 270$ executions parameter space reduction may take place. Thus the improvement of the minimum around the 400-th execution of BACKADD/SEQ may be caused by reducing the parameter space, because the probability to find a better solution has increased. Such a result was observable in most of the instances which we have analyzed.

We consider the instances provided in Table 5 to analyze the influence on computation time of BACKADD and BACKADD/SEQ by increasing the number of items (CPU1 to CPU3), the number of macro-periods (CPU4, CPU1, CPU5), or the number of sub-periods per macro-period (CPU6, CPU1, CPU7). Note, that the other input parameters of INST(J,T^m,T^s,hc,TBO,ΔTBO,dp,cp,U;1) are in all instances as follows: hc = 1, TBO = 25, ΔTBO = 0, dp = (ud,i), U = .8, cp = co.

TABLE 5:
Computation time in sec for # exc = 1000 executions

instance	J	T^m	T^s	BACKADD	BACKADD/SEQ
CPU1	3	10	10	8.4	8.68
CPU2	6	10	10	23.99	23.18
CPU3	9	10	10	46.36	42.95
CPU4	3	5	10	5.5	5.71
CPU5	3	15	10	11.48	11.2
CPU6	3	10	5	9.06	9.23
CPU3	3	10	15	9.17	9.39

The computation time increases substantially (but almost linearly) with the number of items (cf. CPU1 to CPU3 in Table 5), less strongly with the number of macro-periods (cf. CPU4, CPU1, CPU5 in Table 5), and insignificant with the number of sub-periods (cf. CPU6, CPU1, CPU7 in Table 5). The integration of the sequential test requires insignificant computation time. Determining a changeover, i.e. computing the regret measures of the items, requires most of computation time. Thus if there are parameter values which lead to a solution with a lot of changeovers, the computation of such a solution requires a lot of time due to the high number of regret-calculations. If such parameter values are eliminated by the sequential test the computation time per schedule decreases. This may be the reason for a shorter computation time in CPU2 and CPU5 comparing BACKADD/SEQ and BACKADD.

7. Summary and Future Work

The proportional lotsizing and scheduling problem (PLSP) dealt with in this paper is based on the fundamental assumption, that at most one changeover is allowed within each period. Thus the PLSP seems to be a suitable model for supporting short term decisions within an MRP framework.

The backward algorithm BACKADD for solving the PLSP introduced in this paper incorporates several components: Lotsizing and scheduling is done *backward* oriented starting in (the final) period T. In each period t at most two items j are *added. Regrets*

ρ_{jt} are used in a *randomized* way in order to decide which item(s) to produce in a specific period. In addition, the parameter space on which the randomization scheme relies is partitioned. Appropriate subspaces are choosen via hypothesis testing / sequential analysis.

The local search method BACKADD (a Monte Carlo randomized-regret-based biased random sampling method) is able to produce near-optimal solutions within a minimum amount of computational time. Moreover, the ideas introduced in this paper are general enough to become genuine parts of stochastic algorithms.

Until now only very little research has been published on lotsizing (and scheduling) with setup costs and/or times, see e.g. **Cattrysse, Salomon, Kuik, Van Wassenhove** 1990, **Diaby, Bahl, Karwan, Zionts** 1992, **Fleischmann** 1992, **Salomon** 1991, **Kroon, Kuik, Van Wassenhove** 1991 as well as **Trigeiro, Thomas, McClain** 1989. The same holds true for lotsizing (and scheduling) in multi-stage systems, see e.g. **Billington, McClain, Thomas** 1986, **Kuik, Salamon** 1990, **Kuik, Salomon, Van Wassenhove, Maes** 1993, **Maes, McClain, Van Wassenhove** 1991, **Roll, Karni** 1991 as well as **Tempelmeier, Helber** 1992. Therefore in the future it should be worthwhile to investigate the following generalizations of the PLSP (see the models provided in **Drexl, Haase** 1992):

- Setup times should be incorporated into the PLSP in order to ensure feasibility of generated schedules, i.e the PLSP/st should be considered. Especially generalizations of BACKADD (based on modified randomized-regret-based biased random sampling methods with controlled search in the parameter space) should be developed.
- Multiple machines as well as multiple stages have to be incorporated, i.e the PLSP/mm as well as the PLSP/ms should be considered. Once more generalizations of BACKADD should be investigated.

Acknowledgement: The authors are indebted to Erwin Pesch, Rijksuniversiteit Limburg (Maastricht), for helpful comments and suggestions concerning section 5.

References

Billington, P.J., McClain, J.O., Thomas, L.J., "Heuristics for multilevel lot-sizing with a bottleneck", *Management Science*, Vol. 32 (1986), pp. 989-1006.

Bertocchi, M., Brandolini, L., Slominski, L., Sobczynska, J., "A Monte-Carlo approach for 0-1 programming problems", *Computing*, Vol. 48 (1992), pp. 259-274.

Campbell, G.M., Mabert, V.A., "Cyclical schedules for capacitated lot sizing with dynamic demands", *Management Science*, Vol. 37 (1991), pp. 409-427.

Carreno, J.J., "Economic lot scheduling for multiple products on parallel identical processors", *Management Science*, Vol. 36 (1990), pp. 348-358.

Cattrysse, D., Salomon, M., Kuik, R., Van Wassenhove, L.N., "A dual ascent and column generation heuristic for the discrete lotsizing and scheduling problem with setup times", *Management Science*, Vol. 39 (1993), pp. 477-486.

Chen, W.-H., Thizy, J.-M., "Analysis of relaxations for the multi-item capacitated lot-sizing problem", *Annals of Operations Research*, Vol. 26 (1990), pp. 29-72.

Dammeyer, F., Voß, S., "Dynamic tabu list management using the reverse elimination method", Discussion Paper, TH Darmstadt 1992 (*Annals of Operations Research*, to appear).

Dell'Amico, M., Trubian, M., "Applying tabu-search to the job-shop scheduling problem", Discussion Paper, Politecnico di Milano 1992 (*Annals of Operations Research*, to appear).

Diaby, M., Bahl, H.C., Karwan, M.H., Zionts, S., "A Lagrangean relaxation approach for very-large-scale capacitated lot-sizing", *Management Science*, Vol. 38 (1992), pp. 1329-1340.

Dixon, P.S., Silver, E.A., "A heuristic solution procedure for the multi-item, single-level limited capacity, lot-sizing problem", *J. of Operations Management*, Vol. 2 (1981), pp. 23-39.

Dobson, G., "The economic lot-scheduling problem: Achieving feasibility using time-varying lot sizes", *Operations Research*, Vol. 35 (1987), pp. 764-771.

Dorndorf, U., Pesch, E., "Evolution based learning in a job shop scheduling environment", Research Memorandum, Limburg University, Maastricht 1992 (*Computers & Operations Research*, to appear).

Drexl, A., "Scheduling of project networks by job assignment", *Management Science*, Vol. 37 (1991), pp. 1590-1602.

Drexl, A., Haase, K., "A new type of model for multi-item capacitated dynamic lotsizing and scheduling", Discussion Paper, Christian-Albrechts-University of Kiel 1992.

Dueck, G. , Scheurer, T., "Treshold accepting: a general purpose optimization algorithm", *J. of Computational Physics*, Vol. 90 (1990), pp. 161-175.

Elmaghraby, S.E., "The economic lot scheduling problem (ELSP): Review and extensions", *Management Science*, Vol. 24 (1978), pp. 587-598.

Eppen, G.D., Martin, R.K., "Solving multi-item capacitated lot-sizing problems using variable redefinition", *Operations Research*, Vol. 35 (1987), pp. 832-848.

Faigle, U., Kern, W., "Some convergence results for probabilistic tabu search", *ORSA J. on Computing*, Vol. 4 (1992), pp. 32-37.

Fleischmann, B., "The discrete lot-sizing and scheduling problem", *European J. of Operational Research"*, Vol. 44 (1990), pp. 337-348.

Fleischmann, B., "The discrete lot-sizing and scheduling problem with sequence-dependent setup costs", Discussion Paper, University Augsburg 1992 (*European J. of Operational Research*, to appear).

Glover, F., "Tabu search - Part I", *ORSA J. on Computing*, Vol. 1 (1989), pp. 190-206.

Glover, F., "Tabu search - Part II", *ORSA J. on Computing*, Vol. 2 (1990), pp. 4-32.

Goldberg, D.E., "Genetic algorithms in search, optimization, and machine learning". Reading/Mass. 1989

Haase, K., *"Lotsizing and scheduling for production planning"*, 1993 (in preparation).

Hartmann, L.B., van Hee, K.M., "Application of Markov decision processes to search problems", Paper presented at the 2nd ISDSS Conference, FAW/Ulm, June 1992.

Hertz, A., de Werra, D., "The tabu search metaheuristic: How we used it", *Annals of Mathematics and Artificial Intelligence*, Vol. 1 (1990), pp. 111-121.

Irle, A., *"Sequentialanalyse: Optimale sequentielle Tests"*, Stuttgart 1990.

Johnson, D.S., Aragon, C.R., McGeoch, L.A., Schevon, C., "Optimization by simulated annealing: An experimental evaluation; Part I, graph partitioning", *Operations Research*, Vol. 37 (1989), pp. 865-892.

Johnson, D.S., Aragon, C.R., McGeoch, L.A., Schevon, C., "Optimization by simulated annealing: An experimental evaluation; Part II, graph coloring and number partitioning", *Operations Research*, Vol. 39 (1991), pp. 378-406.

Johnson, D.S., Papadimitriou, C.H., Yannakakis, M., "How easy is local search ?", *J. of Computer and System Sciences,* Vol. 37 (1988), pp. 79-100.

Karmarkar, U.S., Schrage, L., "The deterministic dynamic product cycling problem", *Operations Research,* Vol. 33 (1985), pp. 326-345.

Kuik, R., Salomon, M., "Multi-level lot-sizing problem: Evaluation of a simulated-annealing heuristic", *European J. of Operational Research,* Vol. 45 (1990) pp. 25-37.

Kuik, R., Salomon, M., Van Wassenhove, L.N., Maes, J., "Linear programming, simulated annealing and tabu search heuristics for losizing in botteleneck assembly systems", *IIE Transactions,* Vol. 25 (1993) pp. 62-72.

van Laarhoven, P.J.M., Aarts, E.H.L., Lenstra, J.K., "Job shop scheduling by simulated annealing", *Operations Research,* Vol. 40 (1992), pp. 113-125.

Liepins, G.E., Hilliard, M.R., "Genetic algorithms: Foundations and applications", *Annals of Operations Research,* Vol. 21 (1989), pp. 31-58.

Maes, J., McClain, J.O., Van Wassenhove, L.N., "Multilevel capacitated lotsizing complexity and LP-based heuristics", *European J. of Operational Research,* Vol. 53 (1991), pp. 131-148.

Martin, O., Otto, S.W., Felten, E.W., "Large-step Markov chains for the TSP incorporating local search heuristics", *Operations Research Letters,* Vol. 11 (1992), pp. 219-224.

Mühlenbein, H., Gorges-Schleuter, M., Krämer, O., "Evolution algorithms in combinatorial optimization", *Parallel Computing,* Vol. 7 (1988), pp. 65-85.

Pesch, E., *"Local search based learning in automated manufacturing",* Habilitationsschrift, TH Darmstadt 1993.

Pochet, Y., Wolsey, L.A., "Solving multi-item lot-sizing problems using strong cutting planes", *Management Science,* Vol. 37 (1991), pp. 53-67.

Roll, Y., Karni, R., "Multi-item, multi-level lot sizing with an aggregate capacity constraint", *European J. of Operational Research,* Vol. 51 (1991), pp. 73-87.

Salomon, M., *"Deterministic lotsizing models for production planning",* Berlin 1991.

Salomon, M., Kroon, L.G., Kuik, R., Van Wassenhove, L.N., "Some extensions of the discrete lotsizing and scheduling problem", *Management Science,* Vol. 37 (1991), pp. 801-812.

Siegmund, D., *"Sequential analysis, tests and confidence intervals",* New York 1985.

Tempelmeier, H., Helber, S., "A heuristic for dynamic multi-item multi-level capacitated lotsizing for general product structures", Discussion Paper, TH Braunschweig 1992 (*European J. of Operational Research,* to appear).

Thizy, J.-M., Van Wassenhove, L.N., "Lagrangean relaxation for the multi-item capacitated lot-sizing problem: A heuristic implementation", *IIE Transactions,* Vol. 17 (1985), pp. 308-313.

Trigeiro, W.W., Thomas, L.J., McClain, J.O., "Capacitated lot sizing with setup times", *Management Science,* Vol. 35 (1989), pp. 353-366.

Yannakakis, M., "The analysis of local search problems and their heuristics", in: Proc. 7th annual symposium on theoretical aspects of computer science. Berlin (Springer) 1990, pp. 298-311.