Available online at www.sciencedirect.com

SCIENCE DIRECT®

Computers & Operations Research 32 (2005) 985–1006

computers & operations research

www.elsevier.com/locate/dsw

Single-point stochastic search algorithms for the multi-level lot-sizing problem

Jully Jeunet[a], Nicolas Jonard[b,*]

[a]*CNRS, LAMSADE, Université Paris IX Dauphine, Place du Maréchal de Lattre de Tassigny, 75116 Paris, France*
[b]*CNRS, CREA, Ecole Polytechnique, 1 Rue Descartes, 75005 Paris, France*

Abstract

Among the most common decisions in manufacturing and distribution companies are probably those regarding Material Requirements Planning. However, that firms are daily confronted with these decisions does not mean they are easy to handle. The multi-level lot-sizing (MLLS) problem is a combinatorial optimization problem which can only be solved optimally within reasonable delays when small instances are considered. This has motivated the search for heuristic techniques achieving a satisfactory balance between computational demands and cost effectiveness. In particular, the MLLS problem has characteristic features that have permitted the development of specific methods: interdependencies exist among stages in the product structure. In this paper, we examine the performance of single point stochastic techniques and compare them to several problem specific algorithms that exist in the literature. A large set of 280 variants of stochastic search algorithms is designed and applied to a variety of problems of small and large size. We find that these techniques, despite their simplicity and the widespread belief that they are generally efficient, only seldom outperform problem-specific algorithms, and when they do so it is usually associated with a much longer execution time. We also exhibit an efficient combination of search and annealing which is found able to produce significant and consistent improvements over problem-specific algorithms.

1. Introduction

Material requirements planning (MRP) is a method for coordinating replenishment decisions for manufactured items, i.e. end-items made of sub-assemblies and component parts. MRP ensures that the right amounts of components are planned at the various levels of manufacturing and at the right time so that demand for finished goods can be met. As such, it only provides feasible solutions to the multi-level lot-sizing (MLLS) problem. Next comes the problem of finding a high quality

* Corresponding author. Tel.: +33-1-55-55-81-97.
E-mail addresses: jeunet@lamsade.dauphine.fr (J. Jeunet), jonard@poly.polytechnique.fr (N. Jonard).

0305-0548/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2003.09.009

solution, i.e. a sequence of replenishment quantities having the lowest possible cost. This turns out to be a difficult combinatorial problem, especially when real-size set-ups (involving several hundreds of components) are considered.

Much of the early work on the MLLS problem has focused on optimizing algorithms tailored for specific variants. However, as the amount of calculation required for exact solution dramatically increases with the problem size, these optimal methods tend to fall short in even moderate-sized cases. Their computational burden (not to mention their complexity) has motivated the development of numerous specific heuristic procedures.

In this paper we test the performance of stochastic search algorithms on the MLLS problem. Our focus here is on single point stochastic search algorithms, and how well they perform compared to problem specific heuristics. A typical and well known example is simulated annealing, a Monte Carlo method inspired by the way liquids freeze or metals crystallize in the process of annealing (see Metropolis et al. [1] for the general principle and Kirkpatrick et al. [2] for what is probably the first application to combinatorial problems). Although heuristics of that sort are often advertised to be efficient for combinatorial optimization, it will be shown here that a large collection of stochastic single points algorithms is usually outperformed by problem-specific algorithms in spite of the additional computational effort that stochastic search can demand. This result tends to stand in contrast with the traditional view (see for instance Kuik and Salomon [3] on the uncapacitated MLLS problem or Özdamar and Barbarosoglu [4] and [5] on capacitated MLLS). Here, we find that only in large instances significant cost improvements can be observed, and this only obtains with sophisticated transition rules and when the standard (geometric) annealing schedule is abandoned to let the control parameter evolve randomly. As sub-cases of the set of 280 variants presented in the paper, four well-known single-point stochastic algorithms are included: simulated annealing, simulated tempering (Marinari and Parisi [6]), threshold accepting (Dueck and Scheuer [7]) and hill climbing.

The paper is organized as follows. Section 2 is dedicated to the presentation and mathematical formulation of the MLLS problem, together with a short overview of the literature. Section 3 gives the building blocks of the simulated annealing approach, while Section 4 presents the experimental framework. Numerical results are discussed in Section 5 and in Section 6 conclusions and practical implications are derived.

2. The multi-level lot-sizing problem

In manufacturing production systems end-items are made up with a number of intermediate products themselves resulting from combinations of components (until the final stage of purchased parts and raw materials). The bill of materials (the recipe for the end-item) can be depicted as a directed acyclic graph $G(\mathbf{P},\Gamma)$ where $\mathbf{P}=\{1,\dots,P\}$ is the set of vertices (items) and Γ is the set of directed edges symbolizing component-to-product relationships. The existence of a directed edge $(i\to j)\in\Gamma$ means that item i is directly required to assemble j. Item i is then fully defined by Γ_i^{-1} and Γ_i, the sets of its immediate predecessors and successors, together with technical coefficients $c_{i,j}$ that indicate the number of units of i required to produce one unit of j. The set of ancestors of item i is denoted $\hat{\Gamma}_i^{-1}$. Items are sorted in increasing level codes so that any directed edge $(i\to j)$ in the graph is such that $i>j$. Each common part is listed at the lowest level of use in the product

Table 1
Notations for the MLLS problem

Cost parameters	
s_i	Set-up cost for item i (constant over time)
h_i	Unit inventory carrying cost per period for item i
Quantity variables	
$d_{i,t}$	Requirements for item i in period t
$x_{i,t}$	Delivered quantity of item i at the beginning of period t
$I_{i,t}$	Level of inventory for item i at the end of period t
$y_{i,t}$	Boolean variable addressed to capture i's set-up cost in period t
Technical coefficients	
l_i	Lead time for item i,
$c_{i,j}$	Quantity of item i required to produce one unit of item j (production ratio)

structure. Adopting the convention that finished goods belong to level 0, items are then numbered from 1 to P, starting the labeling from level 0 and moving sequentially to the lowest level N.

The MLLS problem consists in finding a sequence of lot sizes that minimizes the sum of set-up and inventory carrying costs, while meeting the demand for end-items over a T-period planning horizon. The notation used to describe the problem is summarized in Table 1.

Cost parameters are time-invariant. This assumption is consistent with most real settings as planning horizons usually do not exceed one year. The optimal solution to the MLLS problem is obtained by minimizing

$$\sum_{i \in \mathbf{P}} \sum_{t=1}^{T} s_{i,t}\ y_{i,t} + h_{i,t}\ I_{i,t}, \tag{1}$$

subject to the set of constraints

$$I_{i,t} = I_{i,t-1} + x_{i,t} - d_{i,t}, \quad \text{for all } i \in \mathbf{P}, \tag{2}$$

$$d_{i,t} = \sum_{j \in \Gamma(i)} c_{i,j}\ x_{j,t+l_j}, \quad \text{for all } i \text{ such that } \Gamma_i \neq \{\varnothing\}, \tag{3}$$

$$x_{i,t} \leqslant M\ \ y_{i,t}, \quad \text{with } y_{i,t} \in \{0,1\}, \tag{4}$$

$$I_{i,t} \geqslant 0 \quad \text{and} \quad x_{i,t} \geqslant 0. \tag{5}$$

The objective function in Eq. (1) is the sum of purchase and production costs, together with set-up and inventory holding costs, for all items over the planning horizon. Eq. (2) is the flow conservation constraint for item i. It defines the inventory level for item i at the end of period t. The initial level of inventory of item i is $I_{i,0}$, which will be set to zero without loss of generality. Requirements consist in the external demand when end-items are considered, and result from the lot sizes of immediate successors for component items, as stated in Eq. (3). We assume that no component is sold to an outside buyer, i.e. external demands only exist for finished goods. Constraint (4), with M a very large number, captures the fact that a set-up cost is incurred each time a batch is purchased or produced. Finally, Constraint (5) states that backlog is not allowed, production being either positive or zero.

Solution methods to that problem fall into five categories; we briefly review them in turn.

Optimal methods. Zangwill [8,9] develops a dynamic programming algorithm for serial systems (each item has at most one predecessor and one successor); Crowston and Wagner [10] consider assembly structures, while Steinberg and Napier [11] examine general ones and formulated the problem as a network model for which the resultant mixed integer problem is solved with a standard solver; Afentakis et al. [12] combine Lagrangian relaxation and branch-and-bound for assembly structures, and Afentakis and Gavish [13] convert complex product structures with component commonality into expanded assembly structures; Saydam et al. [14] use a pure integer programming formulation.

Sequential approaches. Single level lot sizing rules are applied one stage at a time, proceeding down the product structure (Veral and LaForge [15]; Yelle [16] apply different single-stage techniques at each level sequentially). Few general conclusions exist regarding the choice of the appropriate rule under given circumstances; the effect of mixing different lot sizing rules at different levels is also unclear (on that point, see Yelle [16]). Though this approach is pretty natural and simple, it is likely to yield substantial cost inefficiencies for at least two reasons: (i) apart from Wagner and Whitin ([17], WW thereafter), single level rules cannot guarantee an optimal solution for each item individually in the product structure; (ii) the level-by-level approach does not recognize the impact of lot-sizing decisions for a certain item on that item's components.

Cost modification procedures. Modifying costs is a way of achieving some degree of inter-level coordination. Set up costs are altered to account for the fact that placing an order for a certain item can trigger orders for items at deeper levels. Similarly holding costs are altered following the observation that the more an item is stored, the less this item is ordered, hence the lumpier the demand for its components is, and the less these components are stored (Blackburn and Millen [18]). Modified costs are then used in conjunction with sequential application of single-level methods. Bookbinder and Koch ([19], thereafter) have provided an adaptation based on a decomposition scheme for general structures into assembly sub-structures on which the cost modification may be applied.

Multiple passes and cost modifications. In multi-pass procedures the lots for each item are considered several times until no improvement of the solution can be reached. Graves [20] sequentially applies WW to a recursively modified-cost problem. This algorithm starts by determining lot sizes level-by-level. It then computes the marginal cost of meeting one additional unit of demand in each period for item *i*. It finally replaces the original unit cost of *i* by a modified cost including this marginal cost for *i*'s components.[1] Next WW is applied again. If production plans converge the next item is then considered until convergence obtains at each level. Heinrich and Schneeweiss [21] have proposed a two phases multi-pass procedure. The first phase is a multi-pass algorithm to determine the ordering cycles for each item. First, nested solutions for the stationary problem are determined (a constant demand is assumed). Reorder intervals are then increased or decreased until no improvement of the solution is obtained. Phase two uses these ordering cycles to derive modified set-up and holding costs. A single-level technique is then sequentially applied to the cost modified problem; best results obtain when WW is used.

[1] This captures the impact of changes in the schedule for item *i* on lot-sizing decisions for its components.

Period-by-period algorithms. In essence, this class of algorithms proceeds period by period rather than level by level. The lot sizes are simultaneously determined for all stages but for one period at a time. Heuristics based on this approach have been suggested by Lambrecht et al. for serial systems [22] and later for assembly structures [23]. Afentakis [24] has also designed an efficient algorithm for assembly systems. For general structures, however, no adaptation of these algorithms has been made.

Beside these problem specific techniques, a number of general purpose stochastic methods have been applied over the recent years. Kuik and Salomon [3] have applied the simulated annealing search method to the uncapacitated MLLS problem. From their experiments, the authors conclude that good solutions can be found by simulated annealing in reasonable computation times only for small problems. Consequently, they suggest incorporating more problem-specific features in the design of an efficient simulated annealing search method. Dellaert et al. [25] have developed an efficient genetic algorithm to solve the MLLS problem with time-varying costs, a problem type for which few heuristics are available. In Dellaert and Jeunet [26], a hybrid genetic algorithm is showed to address very competitively large-scaled problems with component commonality in a context of constant cost parameters.

3. Implementing stochastic search

In this section, we first discuss the issues related to problem representation. We then provide a schematic description of the single-point stochastic algorithms that will be implemented.

3.1. Problem representation and feasibility

A natural encoding is to represent the list of delivery dates for all items as a $P \times T$ binary matrix $\mathbf{Y} = (y_{i,t})$, with $y_{i,t} = 1$ if an order for item i is delivered in period t and $y_{i,t} = 0$ otherwise. Denote $\mathbf{\Omega}$ the set of all possible $P \times T$ binary matrices. Searching for an optimal solution to the MLLS problem amounts to finding $\mathbf{Y}^* \in \mathbf{\Omega}$ such that the corresponding quantities minimize the objective function (1).

The vector $y_i = (y_{i,1}, \dots, y_{i,T})$ coding delivery dates for item i is called a string. For any binary matrix, the decoding—that is the conversion of delivery dates into lot sizes—is performed by moving sequentially from finished goods to purchased items, translating final product requirements into component part requirements. To illustrate, assume $\mathbf{P} = \{1, 2\}$ and $\Gamma = \{(2 \to 1)\}$. All lead times are set to one and a one-to-one production ratio is assumed ($c_{2,1} = 1$). The horizon length equals 6 periods ($T = 6$). For a stream of final demand $d_1 = (0, 0, 1, 5, 10, 2)$ and a binary matrix $\mathbf{Y} = (y_1, y_2) = ((0, 0, 1, 0, 1, 0), (0, 1, 0, 0, 0, 0))$, we get $x_1 = (0, 0, 6, 0, 12, 0)$ yielding $d_2 = (0, 6, 0, 12, 0, 0)$ the requirements for component 2, and using y_2 we obtain $x_2 = (0, 18, 0, 0, 0, 0)$.

The rationale for such a binary encoding is that optimal solutions to the concave cost problem considered here must satisfy

$$x_{i,t} \cdot I_{i,t-1} = 0, \quad \forall i \text{ and } t. \tag{6}$$

Property (6) simply states that optimal lot sizes always cover an integer number of future demands or, put another way, it is never optimal to launch more than one order to satisfy the demand of one period (see Veinott [27]). Indeed, for an item i such that $d_{i,t} > I_{i,t-1} > 0$, an extra lot has to be received in period t to meet demand (implying $x_{i,t} \cdot I_{i,t-1} > 0$). This means that not only carrying costs are incurred, but also that the set-up cost is payed twice.

Regarding the feasibility of candidate solutions, let $p_{i,k}$ be the kth period of positive requirements for item i, $k = 1,\dots,K$ and $K \leqslant T$. First note that necessarily $y_{i,t} = 0$ for $t < p_{i,1}$ and $y_{i,p_{i,1}} = 1$. This is simply restating that there is no production before the first requirement, and that the first requirement must be covered. The same holds after the last requirement. Finally, between any two consecutive requirements there has to be at most one order. Formally, this is written as

$$\begin{aligned}
&y_{i,t} = 0, \quad \text{for } t = 1,\dots,p_{i,1}-1,\\
&y_{i,t} = 1, \quad \text{for } t = p_{i,1},\\
&y_{i,t} = 0, \quad \text{for } t = p_{i,K}+1,\dots,T,\\
&\sum_{t=p_{i,k}+1}^{p_{i,k+1}} y_{i,t} \leqslant 1, \quad \text{for } k = 1,\dots,K-1.
\end{aligned} \tag{7}$$

A matrix $\mathbf{Y}$ is said to be feasible if and only if, for all $i \in \mathbf{P}$, the previous conditions are met.

3.2. General principles

At each iteration a single point stochastic algorithm explores a neighborhood of the current solution point $\mathbf{Y}$. A neighbouring candidate solution $\mathbf{Z}$ is generated by means of a transition rule, which for instance could change ‘0’s into ‘1’s or the opposite at some places in $\mathbf{Y}$. The resulting candidate $\mathbf{Z}$ is evaluated through a cost function $C : \mathbf{\Omega} \to \Re^+$. In our case, if $\mathbf{Z}$ is feasible—that is if $\mathbf{Z}$ meets conditions (7)—the decoding procedure given in Section 3.1 is applied, inventories are computed and the evaluation takes place through $C(\mathbf{Z}) = \sum_{i,t} s_{i,t}\, z_{i,t} + h_{i,t}\, I_{i,t}$, the objective function (1). If $\mathbf{Z}$ is not feasible, the cost function returns $C(\mathbf{Z}) = +\infty$ unless a repair procedure is employed to make $\mathbf{Z}$ feasible. Acceptance of $\mathbf{Z}$ as the new current solution depends on the cost differential between $\mathbf{Z}$ and $\mathbf{Y}$: generally the higher $C(\mathbf{Z}) - C(\mathbf{Y})$ is, the lower the odds for $\mathbf{Z}$ to replace $\mathbf{Y}$, and this is governed by a control parameter. When the termination condition is met—generally after several updatings of the control parameter—the best solution found over the entire search is returned. Fig. 1 summarizes the main steps of the procedure.

Obviously many degrees of freedom exist regarding the design of the acceptance mechanism, the evolution of the control parameter, the termination condition, the rule for neighbourhood search or the way of addressing infeasibility. These points are discussed in detail below.

3.3. Acceptance probability

Any improving move should be accepted with probability one. Differences can arise in the treatment of bad—that is to say cost deteriorating—candidates. The two major threads consist in either using a smooth probabilistic function of the cost difference, or a step function.

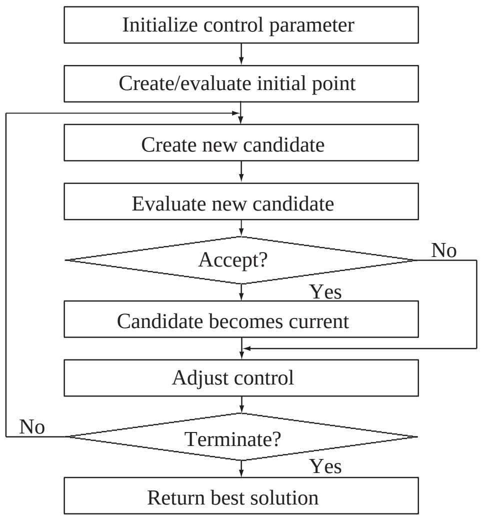

Fig. 1. The general structure of a single point algorithm.

In simulated annealing (SA) a cost-deteriorating move is accepted with a probability that decreases continuously with the cost differential and depends on the value of a control parameter $\theta > 0$ that is usually interpreted as the temperature. Time is divided into epochs of identical length ℓ. Epochs are indexed with $e = 1, 2, \ldots$ and time is indexed with $s = 1, 2, \ldots$. Within an epoch the transition rule is called ℓ times and the current solution point changes at most ℓ times. At each point s in time, with $e \equiv \lfloor s/\ell \rfloor$, a candidate $\mathbf{Z}$ replaces the current point $\mathbf{Y}_s$ with probability

$$\pi_s = \Pr\{\mathbf{Y}_{s+1} = \mathbf{Z}/\mathbf{Y}_s\} = \begin{cases} 1 & \text{if } C(\mathbf{Z}) \leqslant C(\mathbf{Y}_s), \\ \exp\{-[C(\mathbf{Z}) - C(\mathbf{Y}_s)]/\theta_e\} & \text{otherwise.} \end{cases} \tag{8}$$

For a given cost differential, a high temperature (a large θ) entails a high probability of uphill moves (worse solutions) whereas low temperatures yield low acceptance probabilities for worse candidates.

In threshold annealing (TA), by contrast, a move which makes the current solution worse than a predefined threshold is never accepted: the probabilistic model (8) from SA is simplified to a deterministic threshold rule. This threshold is gradually decreased during search like temperature in SA. A candidate $\mathbf{Z}$ replaces the current point $\mathbf{Y}_s$ with probability

$$\pi_s = \Pr\{\mathbf{Y}_{s+1} = \mathbf{Z}/\mathbf{Y}_s\} = \begin{cases} 1 \text{ if } C(\mathbf{Z}) \leqslant C(\mathbf{Y}_s) + \theta_e, \\ 0 \text{ otherwise.} \end{cases} \tag{9}$$

When it is assumed that $\theta_e = 0$ for all $e \geqslant 0$ in (9) or $\theta_e \approx 0$ for all $e \geqslant 0$ in (8), the resulting algorithm is simply performing hill climbing (HC). HC is therefore a particular case of both SA and TA. Like TA, HC introduces determinism in search but unlike TA, worse solutions are never explored; this makes HC more vulnerable to premature entrapment in local minima.

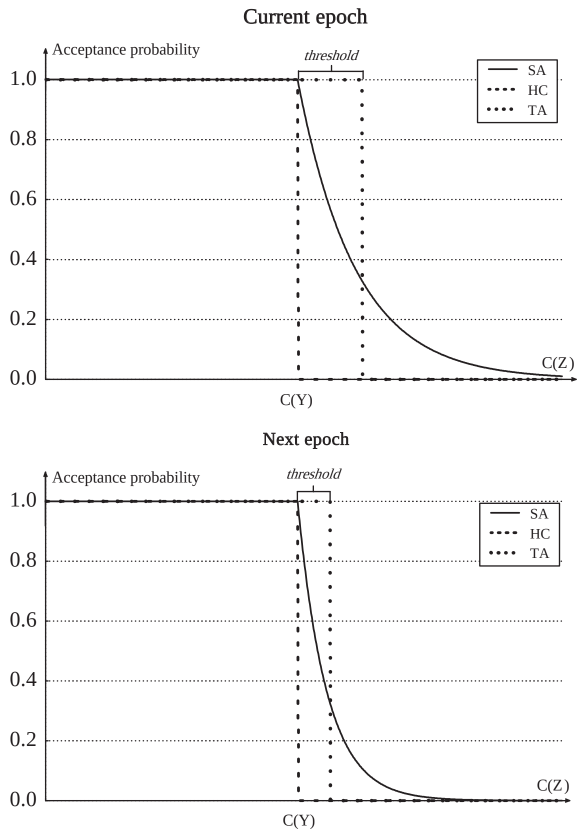

Fig. 2. Acceptance probability at epoch e and $e+1$ as a function of the candidate cost values $C(\mathbf{Z})$, for SA, HC and TA.

As an illustration, Fig. 2 plots the acceptance probability at two consecutive epochs for SA, TA and HC.

3.4. Changing the control parameter: cooling schedule

A common feature of the algorithms included in this study is that there is a parameter controlling how easily cost-deteriorating moves are accepted. In SA this parameter is the temperature θ, which stays constant over epochs of lengths ℓ, and then changes. The control parameter is assumed to take its values in a finite (possibly very large) set $\mathbf{\Theta} = \{\theta^{(0)}, \ldots, \theta^{(m)}\}$, that without loss of generality is supposed to be ordered: $\theta^{(0)} > \cdots > \theta^{(m)}$. SA demands that θ approaches 0 as time passes, so for a given cost differential acceptance is less and less likely and the system eventually freezes in a minimum of the cost function. By contrast, in simulated tempering (ST) the temperature parameter is treated stochastically: θ randomly wanders within $\mathbf{\Theta}$. It is also possible that the set $\mathbf{\Theta}$ evolves endogenously as in van Laarhoven and Aarts [28]. The rule that dictates the changes in the control parameter θ from one epoch to the next is called the cooling schedule. Three cooling schedules are considered in this study: proportional cooling, tempering, and endogenous cooling.

3.4.1. *Exogenous evolution: proportional cooling and tempering*

In proportional (or geometric) cooling the set of temperature values $\mathbf{\Theta}$ is generated by choosing an initial value $\theta^{(0)}=\theta_0$ and assuming that $\theta^{(r+1)}=\alpha\,\theta^{(r)}=\alpha^{r+1}\theta_0$ for all $r \geqslant 1$. Parameter $0<\alpha<1$ is called the scale parameter. Proportional cooling then simply assumes that given $\theta_e=\theta^{(r)}$, one has

$$\theta_{e+1}=\theta^{(r+1)}. \tag{10}$$

Hence starting with θ_0, the temperature is geometrically updated after a fixed number ℓ of (proposed) transitions Kirkpatrick et al. [2].

Assume now the set $\mathbf{\Theta}$ is generated exactly like in proportional cooling but the process of temperature change is a symmetric random walk over $\mathbf{\Theta}$. This procedure is employed in ST. Given $\theta_e=\theta^{(r)}$ the value of the control in epoch e, one has

$$\theta_{e+1}=\begin{cases}\theta^{(r-1)} \text{ with probability } 1/2,\\ \theta^{(r+1)} \text{ with probability } 1/2,\end{cases} \tag{11}$$

with the proper boundary conditions. Temperature at each epoch can be either increased or decreased. By warming up in some epochs, the algorithm is able to escape local minima and possibly approach a global one. The flaw is that nothing prevents the algorithm from moving far away again from the global optimum (more in Marinari and Parisi [6]).

3.4.2. *Endogenous evolution*

Contrary to the previous schedules where change is exogenous, it is possible to introduce a feedback so that the control parameter is sensitive to the success of exploration. The rule proposed by van Laarhoven and Aarts assumes cooling is not governed anymore by a set $\mathbf{\Theta}$ of temperature values. Rather, given an initial value θ_0 it is assumed that

$$\theta_{e+1}=\frac{\theta_e}{1+\theta_e(\ln(1.1)/3\sigma_e)}, \tag{12}$$

where σ_e is the standard deviation of the cost of accepted moves during epoch e. The larger σ_e is, the more it is expected that search produces improvements, and therefore the smallest the parameter change. By contrast, when σ_e is close to 0, the change in θ change can be quite dramatic as prolonging exploration does not seem fruitful.

3.5. *Terminating search*

The stopping criterion used in the experiment has an upper bound to gain control over the CPU expense, namely that search is no longer than 50 epochs. For geometric and endogenous cooling it has the extra feature that search stops at the beginning of epoch $e+1$ if

$$\theta_{e+1}>\frac{\sigma_e}{10^{-3}(\bar{C}_1-\bar{C}_e)}, \tag{13}$$

with $\bar{C}_e$ the average cost in epoch e. Finally in the case of a tempering scheme, the algorithm is forced to work as long as geometric cooling on average does, as there is no natural termination criterion. Hill climbing has a similar problem, so it was decided to stop it if 3 or less improvements are reached in an epoch. During search the best solution found is stored, and returned when search terminates.

3.6. Transition rules

Given the current point $\mathbf{Y}$, how is a candidate point $\mathbf{Z}$ generated? A natural principle could be to alter a few randomly chosen bits in $\mathbf{Y}$. All the rules that are considered obey that logic, but differ in terms of how random modifications propagate along the stages of inventory.

R_1. The first rule is one of the two best performing ones in the study by Kuik and Salomon [3]. Given the current point $\mathbf{Y}$, the transition rule R_1 randomly selects a new candidate point $\mathbf{Z}$ by randomly selecting a pair (i^*, t^*) and switching y_{i^*,t^*} to $1 - y_{i^*,t^*}$. Such a single-bit search generates offsprings that most of the time violate the feasibility constraints. Two strategies to address this problem are discussed in the next subsection.

R_2. The second rule searches the same domain with the additional constraint that each bit change takes place one level higher in the product structure (it is the second most efficient rule in the study by Kuik and Salomon [3]). Letting $n(i)$ be the level code of the last mutated item i, rule R_2 randomly selects a pair (i^*, t^*) with $n(i^*) = n(i) + 1$ and switches y_{i^*,t^*} to $1 - y_{i^*,t^*}$. When the deepest level of the structure is reached, search starts again at level 0. The rule is slightly more problem specific as it recognizes that a change at one level of the product structure can trigger changes below, so it can be a good thing to force mutations to proceed down the structure. Feasibility is again a problem.

R_3. Rule R_3 extends the previous ones by implementing a structure-wide generalization of the single bit mutation. One mutation is performed first, by choosing (i^*, t^*) and changing y_{i^*,t^*} to $1 - y_{i^*,t^*}$. Then the same mutation is performed (in a probabilistic manner) on all the ancestors of i^* with the appropriate lead time correction. This all-ancestor cumulative mutation accounts for product interdependencies across all the inventory. More precisely the procedure is the following. Let $j \in \hat{\Gamma}_{i^*}^{-1}$ be an ancestor of i^* and let $h \in \Gamma_j$ be a successor of j. Set y_{j,t^*-l_h} equal to y_{h,t^*} if either (1): h has been mutated earlier in the cumulative process and j has never been mutated at this position, or (2): j has already been mutated at this position, in which case j has a probability of being changed equal to $1/|\Gamma_j|$ (decreasing with the number of successors of j). The rationale is that first the procedure avoids undoing what has been done as much as possible. Second, the likelihood that the demand and in turn the lot sizes for j have to be changed in response to a change in the lot sizes for h decreases with $|\Gamma_j|$. Indeed, in the extreme case of a unique h, it is obvious that a mutation for him entails a change for j, whereas when many hs exist there is no reason to systematically reproduce the mutation on j.

R_4. Rule 4 is a cumulative mutation restricted to immediate ancestors. When a mutation is performed on item i^* at time t^*, the same mutation is performed on any $j \in \Gamma_{i^*}^{-1}$ with the appropriate lead time correction. This immediate-ancestor cumulative mutation is less disruptive than the previous as changes have a more localized search, as opposed to the sort of global search performed by R_3.

R_5. The last rule is a cumulative insertion/cancellation restricted to immediate ancestors. The first step is a choice between insertion and cancellation of an order with probability 1/2. The procedure then selects an item i^* and an appropriate date to perform the operation. Then for the item and all its predecessors (any $j \in \Gamma_{i^*}^{-1}$) an order is either inserted or cancelled between two consecutive dates of positive requirements.

The transition rule is central in designing an efficient search algorithm. Simple rules might prove inefficient in a problem exhibiting as many interdependencies as the MLLS problem does.

3.7. Feasibility: penalizing or repairing

A candidate point in $\boldsymbol{\Omega}$ does not necessarily satisfy the feasibility constraints as they are stated in (7). Rather, it is often the case that infeasibility arises when random mutations take place. At that stage either a very large cost is associated to an infeasible candidate (as in Kuik and Salomon [3]), or the candidate undergoes repairing. Each option has its shortcomings. The first one produces strings with infinite costs that are never accepted, though they are still evaluated and therefore consume CPU. The second option forces corrections in different places of the candidate, which is also CPU consuming. [2]

Penalizing. A specific routine is used to compute the cost of an infeasible candidate; it involves solving an optimization program (Flash Dual in Kuik and Salomon [3]) which returns a cost whatever the feasibility of the candidate. An arbitrarily large cost is then associated with infeasible candidates.

Repairing. The repairing routine simply implements the adequate changes by inserting or cancelling orders to guarantee that each string in the altered chromosome satisfies (7). Generally the closer from end-items the original modification takes place, the more disruptive its consequences are likely to be (hence the more subsequent corrections have to be performed to preserve feasibility).

Both means of coping with infeasibility will be tested.

3.8. Comparison

The decision to replace $\mathbf{Y}$ with $\mathbf{Z}$ depends on the cost difference $C(\mathbf{Z}) - C(\mathbf{Y})$. An alternative formulation that might improve performance but does significantly modify the spirit of the algorithm is to use the best point so far—the minimum cost—rather than the current one. If this option is chosen the acceptance probability schemes given in Section 3.3 should be modified so that $\mathbf{Y}_s$ is replaced with $\underline{\mathbf{Y}}_s = \min_{s' \leqslant s}\{\mathbf{Y}_{s'}\}$. The rationale for this variant is to prevent the system from moving away too quickly from the best point so far by accepting small successive deteriorations; at the same time the flaw is that if a local optima with cost larger than the best cost so far is reached, it is harder to escape than under the standard comparison scheme.

4. Experimental design

In this section we discuss the parameter values and the initialization procedure, and present the test problems.

[2] A third route could be to accept a reasonable amount of infeasible moves so that the system explores regions of the solution space that would remain out of reach if infeasible candidates were systematically precluded. This can be obtained by making the penalty not so large that rejection is certain. This idea is implemented in Gendreau et al. [29] by incorporating in the objective function penalty terms proportionate to the deviation from constraints satisfaction.

4.1. Parameters and initial conditions

Several elements have to be described in order for the algorithms to be fully specified.

Starting point. The quality of the final solution and the speed of convergence to this final solution depends on the starting point. The first problem is therefore the choice of a proper initial solution $\mathbf{Y}_0$. Two options are considered: the solution provided by the best level-by-level technique $\mathbf{Y}_{bestSL}$, and a randomly generated feasible solution having '0's and '1's in equal proportions. The single-level techniques that are employed sequentially in this study are those by Silver and Meal [30], the periodic order quantity (POQ), the economic order quantity (EOQ), Wagner Whitin (WW), least unit cost method (LUC) and part-period balancing (PPB). The Silver–Meal (SM) heuristic selects the order quantity so as to minimize the cost per time unit over the time periods the order lasts. The EOQ is the traditional Wilson formula that balances the inventory carrying costs and the ordering costs (it provides the optimal solution when demand is constant). Periodic Order Quantity uses the EOQ to compute a reorder time cycle and then orders the demand for that time cycle. The WW algorithm provides the optimal solution to the single-level lot-sizing problem by use of dynamic programming. The Least Unit Cost method selects the order quantity so as to minimize the cost per unit (cumulation of the demands until the cost/unit ratio starts to increase). Finally Part Period Balancing includes a future demand in the lot size as long as the additional carrying cost generated by the inclusion of this demand is less than the cost of placing a new order for that period's requirement.

Setting the control parameter. The initial value θ_0 should be such that, at the outset, almost any dis-improvement is accepted. For each variant a random sample of 200 cost variations is generated and used to compute an average cost variation $\bar{\delta}$. An individual cost variation is produced by randomly creating a feasible string and applying the corresponding transition rule. For the exponential acceptance probability, θ_0 is obtained according to $\theta_0 = -\ln(0.95)/\bar{\delta}$, where 0.95 is the probability of accepting the average cost increase. For the step function, $\theta_0 = 0.95\ \bar{\delta}$. Regarding geometric cooling, it is assumed that $\alpha = 0.97$, a value that guarantees a reasonably slow convergence. Similarly there are 10 equally log-spaced temperatures in $\mathbf{\Theta}$ for tempering, ranging from θ_0 to $10^{-3}\ \theta_0$.

Epoch length. The epoch length ℓ is the number of periods over which the value of θ is kept constant, and has length proportionate to the problem size: practically we used $\ell = PT/2$ as in Kirkpatrick et al. [2].

4.2. The test problems

Again problems can be defined along several dimensions. We have opted for a two-round testing procedure, in which we first seek to isolate a sub-group of cost efficient algorithms over a small set of reasonable size problems (40 products and 12 periods), that are then applied to a set of larger problems.

The structures examined consist of a single end-item made of 39 components (hence $P = 40$) distributed over 8 levels (0 for the end-item, and $1,\dots,7$ for components, hence $N = 7$). Product structures are defined in terms of complexity. We use the complexity index proposed by Kimms [31]. Recall products are numbered in topological order by the integers $1,\dots,P$ and let $P(n)$ be the number of products at level $n = 0,\dots,N$ (clearly $P = \Sigma_{n \leqslant N} P(n)$). The most complex—in the sense of having the largest number of product interdependencies—structure is obtained when each item enters the composition of all the items located at higher levels in the product structure. By contrast, the simplest structure obtains when each item enters the composition of one item belonging to a higher level. Kimms defines the complexity of a product structure as

$$C = \frac{A - A_{\min}}{A_{\max} - A_{\min}}, \tag{14}$$

where $A = \Sigma_{i=1}^{P} |\Gamma_i|$ is the actual number of edges in the structure. There is of course a minimal number of arcs such as the product structure is connected, which we denote $A_{\min}$ and is equal to $P - P(0)$. Conversely the graph has an upper bound $A_{\max}$ to the number of edges it can contain: $A_{\max} = \Sigma_{n=0}^{N-1} \{P(n)\ \Sigma_{j=n+1}^{N} P(j)\}$. Structures for which $A = A_{\min}$ are necessarily assembly structures with $C = 0$, whereas structures such as $A = A_{\max}$ satisfy $C = 1$. We chose to take C in the set $\{0, 0.25, 0.5, 0.75\}$. We assumed a one-to-one production ratio (one unit of component is required to produce one unit of its immediate successor, $c_{i,j} = 1$) and set lead times to zero. External demand for the end-item is generated from a truncated normal distribution $\mathcal{N}(50, 25)$ to avoid negative demands. The planning horizon length is $T = 12$ periods.

In accordance with the assumption of value-added holding costs, carrying costs for each item are defined as

$$h_i = e_i + \sum_{j \in \Gamma^{-1}(i)} c_{j,i}\ h_j \quad \text{with } e_i = 0.0005 + 0.02\ u, \tag{15}$$

where $u \sim \mathrm{U}[0, 1]$.

We used the time between orders (TBO) factors to determine set-up cost parameters s_i. TBOs are usually larger for components than they are for end-items and subassemblies. Let τ_1 be the TBO for the end-item and let τ_P be the TBO for the purchased item P. Then τ_P takes the highest value amongst all TBOs in the product structure whereas τ_1 takes the lowest value (end-item 1 has the maximum number of ancestors whereas P has no ancestor). To allocate a meaningful TBO-value to any item i, we need to define τ_i as an increasing function of the number of ancestors $|\hat{\Gamma}_i^{-1}|$ bounded by τ_1 and τ_P. For all items $i \neq 1$, we defined $\tau_i = (\tau_1 - \tau_p)|\hat{\Gamma}_i^{-1}|/|\hat{\Gamma}_1^{-1}| + \tau_p$ and chose two pairs of values $(\tau_1, \tau_P) \in \{(1, 2), (2, 6)\}$. For each item we then set

$$S_i = 0.5\ \bar{d}_i\ h_i\ \tau_i^2 \quad \text{with } \bar{d}_i = \sum_{h \in \Gamma_i} c_{i,h}\ \bar{d}_h. \tag{16}$$

Finally each of the $2 \times 4 = 8$ experiments (2 levels of TBOs, 4 levels of complexities) was replicated 5 times, therefore providing 40 distinct instances.

The single point stochastic algorithms studied here possess 6 defining features as summarized in Table 2, hence a total number of configurations that amounts to $2 \times 3 \times 5 \times 2 \times 2 \times 2 = 240$ specifications. They will all be tested on the set of problems that have been presented.

Of course other options would be available, but we expect that 240 variants will already permit some robust conclusions to be drawn. As the performance of stochastic algorithms is dependent on the sequence of random events that occur in the course of search, 10 runs are performed for each test case. Once efficient settings are identified, they will be applied to a set of larger problems in the second phase. HC corresponds to a step function at $\theta = 0$; it is tested on the same set of problems with the termination criterion that search stops when 3 or less improvements have been obtained over one epoch. The acceptance probability and the evolution of temperature are irrelevant in HC, leaving $5 \times 2 \times 2 \times 2 = 40$ possible specifications.

Table 2
Factor combinations

Factors	Variants	
Acceptance probability	Exponential/step	2
Cooling schedule	Geometric/tempering/endogenous	3
Neighbourhood search	Five transition rules	5
Constraint satisfaction	Penalizing/repairing	2
Comparison	Current/best	2
Initialization	Best single level heuristic/random	2

The second phase of the experiment involves larger product structures embedding 200 items and a longer 24-period planning horizon. Again we set $C \in \{0, 0.25, 0.5, 0.75\}$. We consider a unique end-item together with a one-to-one production ratio, lead times are randomly drawn in $\{0, 1\}$ and the structure depth has 15 levels, a reasonable value given the number of components. Demand in each period for the end item is randomly drawn according to a truncated normal distribution $\mathcal{N}(50, 25)$. We generated 100 tests (replications of costs and demands) for each value of the complexity index C, hence a total of 400 cases was considered in the second phase.

For the purpose of comparison, several problem specific heuristics are included in the study. It has already been mentioned that the best sequential technique is chosen among the economic order quantity, the period order quantity, Wagner and Whitin, the least unit cost method, part-period balancing and Silver and Meal's procedure. To this list we add the lot-for-lot policy, the technique proposed in Bookbinder and Koch, and the multi-pass algorithms of Graves, as well as Heinrich and Schneeweiss which generally produce high quality solutions.

5. Results

This section is devoted to the presentation and discussion of the experimental data produced in the two test phases.

5.1. Phase 1: overall comparison

First factor influence is investigated using the non-parametric Kruskal–Wallis ANOVA for paired samples of cost value. The null hypothesis is always that the samples come from the same distribution, while the factors are those in Table 2 augmented with complexity and the TBO factor. The results are that there is a significant influence on costs for C, the TBO factor, the functional form of the acceptance probability, the cooling schedule, the transition rule, the repairing procedure and the starting point with all critical probabilities less than 10^{-3}. Only the comparison criterion ($p_c = 0.8373$) is non significant.

Table 3 provides averaged cost ratios benchmarked against the whole set of variants. For each problem the comparison is made between each problem specific technique and each of the 240 variants. For instance, the average for the lot-for-lot (L4L) technique in a given problem is computed along

$$r = \frac{1}{240}\sum_{v}\frac{100 \times C(\mathbf{Y}_{\mathrm{L4L}})}{C(\mathbf{Y}_v^*)}, \tag{17}$$

where v stands for a variant, and these ratios are then averaged over the 40 problems. The table should read row by row, and then a low cost ratio suggests an efficient technique. For the table to be read column by column, the interpretation should be reversed. For instance the fact that using a tempering schedule is the most efficient alternative is expressed by having, in any column, the cost ratio of techniques over variants implementing tempering being larger than when geometric or endogenous cooling is adopted (in L4L, $195.1 > 192.9$ and $195.1 > 183.7$). The empty cells in the hill climbing (HC) column correspond to irrelevant factor decompositions.

From Table 3 it is clear that the technique by Heinrich and Schneeweiss outperforms the global pool of stochastic search techniques, whereas the other rules included in this study are themselves outperformed by stochastic search. As intuition would predict, pure hill climbing never outperforms more sophisticated search routines in which worsening moves are accepted. Tempering, i.e. random temperature changes, turns out to be the most promising cooling schedule. Regarding the efficiency of transition rules, it is clear that the all-ancestor cumulative mutation R_3 is by far the best operator. However this global result with averaging over 240 variants hides strong disparities, as will be shown in the second part of the experiment. A first preliminary ranking on the basis of these cost ratios would nonetheless suggest to use an exponential acceptance probability, tempering, the ancestor based cumulative mutation, a repair procedure and the best solution as a starting point.

At a more disaggregated level, it is possible to get graphical insights into the effect of each factor. For each test instance, it is usually the case that more than one variant proposes the same final solution. Within this set, it is possible to sort variants depending on the factor values. In Fig. 3 the proportion of variants embedding the factor value that achieves the best cost is provided for each of the 6 factors, over the 40 test cases. Problems are arrayed on the horizontal axis, the first ten problems corresponding to $C = 0$, the next ten to $C = 0.25$ and so on.

Direct observation of the upper left panel in Fig. 3 suggests comparable efficiency for the exponential acceptance function and threshold acceptance. Testing has nonetheless showed that progressively discarding cost deteriorating candidates is a better option than a more abrupt policy. The upper right panel provides evidence of the poor performance of endogenous cooling as opposed to standard geometric cooling and (perhaps more surprisingly) tempering, which globally outperforms its two rivals. The good performance of tempering stems from its (embedded) ability to escape local minima. However, as the system is not cooling in the standard sense and therefore costs are not converging, an arbitrary number of epochs has to be imposed. This can be somewhat delicate to parametrize ex-ante. Endogenous cooling terminates search faster and happens to be difficult to tune, although for different reasons. The middle left panel in Fig. 3 clearly illustrates the strength of rule R_3 in exploring neighbouring candidates. Rule R_5 also achieves (though to a much less pronounced extent) a reasonable performance in some instances, but R_3 is clearly the best choice. Regarding infeasibility and the way of coping with it, modifying the candidate strings to restore feasibility is the best solution as indicated in the middle right panel. The lower left panel in Fig. 3 suggests that which comparison criterion should be employed is unclear as both appear in successful variants, while finally the lower right panel indicates a clear positive effect of starting search at the solution provided by the best available sequential heuristic. Starting from a random point does not

Table 3
Overall performance and cost ratios by factor level relative to stochastic search, 40×12 test instances

Overall performance Decomposition by factors	Lot sizing rule					
	L4L	*bestSL*	BK	HS	G	HC
	190.6	118.8	125.4	88.4	104.5	104.9
Acceptance						
Exponential	192.1	120.0	126.6	89.3	105.5	—
Step	189.0	117.6	124.2	87.5	103.4	—
Cooling						
Geometric	192.9	120.6	127.3	89.7	106.0	—
Tempering	195.1	121.8	128.5	90.6	107.0	—
Endogenous	183.7	114.1	120.0	84.9	100.3	—
Search rule						
R_1	180.8	112.1	118.4	83.4	98.5	106.0
R_2	173.1	107.4	113.4	80.0	94.5	105.2
R_3	207.2	129.6	136.7	96.4	113.8	104.8
R_4	194.4	121.6	128.3	90.4	106.8	105.0
R_5	197.4	123.5	130.3	91.8	108.5	103.7
Feasibility						
Repair	194.5	121.6	128.3	90.4	106.8	105.9
Penalize	186.7	116.1	122.6	86.4	102.1	104.0
Comparison						
Current	190.7	118.9	125.5	88.5	104.5	105.1
Best-so-far	190.4	118.7	125.3	88.3	104.4	104.9
Initialization						
Random	187.6	116.4	123.0	86.6	102.3	103.9
bestSL	193.5	121.2	127.8	90.3	106.5	106.0

Column headings stand for lot-for-lot (L4L), best sequential heuristic (*bestSL*), Bookbinder and Koch (BK), Heinrich and Schneeweiss (HS), Graves (G) and hill climbing (HC).

necessarily prevent the system from reaching a satisfactory performance, but it is certainly a slower solution.

Regarding execution time, tests were performed on a 1.7 GHz Pentium IV under Windows NT and the programming language was C++. On the set of 40×12 problems, the average execution time for the lot sizing specific techniques is 0.005 CPU s, while stochastic search demands an average of 0.514 s, i.e. more than a hundred times more. The effect of complexity is pretty marked, with 0.371 s for $C = 0$, then 0.407 s for 0.25, 0.526 for 0.5 and finally 0.531 s for $C = 0.75$.

Based on both the observation of average cost ratios and Fig. 3, a reasonable combination of factors to be tested against larger problems can now be proposed. Comparison should be with the current rather than the best-so-far candidate. The standard exponential probability of acceptance

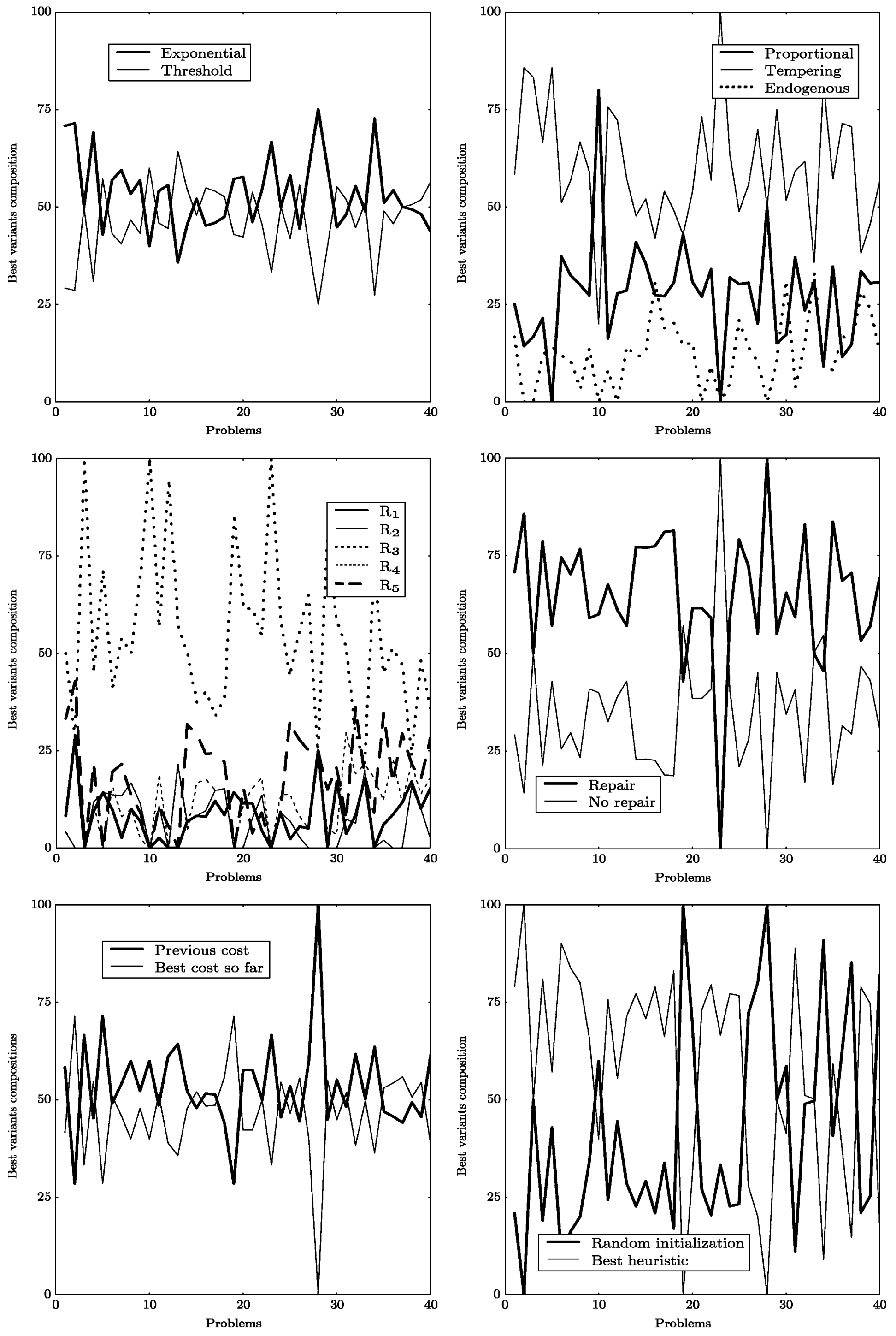

Fig. 3. The efficiency of each of the 6 factors for the 40 test cases.

Table 4
Overall performance and cost ratios by factor level relative to simulated tempering, 200 × 24 test instances

Rule	Overall	Complexity (C)				TBO	
		0	0.25	0.5	0.75	(1,2)	(2,6)
SA	100.3	100.3	101.7	99.8	99.6	99.1	101.7
HC	106.5	105.8	107.7	107.1	105.4	104.6	108.4
HS	107.5	102.1	100.1	111.6	115.9	106.9	107.9
G	138.5	125.7	140.3	142.5	145.3	144.8	132.1
bestSL	157.8	150.6	159.3	159.6	161.6	171.1	144.4
BK	176.1	154.6	180.5	183.4	185.7	174.9	177.2

Row headings stand for simulated annealing (SA), hill climbing (HC), Heinrich and Schneeweiss (HS), Graves (G), best sequential heuristic (*bestSL*) and Bookbinder and Koch (BK).

should be retained, a repairing procedure should be applied on potential candidates, the transition rule should be R_3, the cumulative ancestor-biased mutation, and search should begin with the best solution provided by sequential level-by-level heuristics. As for the evolution of the control parameter θ, both proportional cooling and tempering will be used, hence the algorithms tested will simply be denoted SA and ST.

5.2. Phase 2: efficient algorithm test

In this section the best factor combinations are tested against problem specific heuristics on a set of 400 problems with 200 items over a 24-period horizon. Benchmarking is done with respect to the solution cost proposed by the most efficient technique, ST. This calculation is done for each instance, and averages are taken over factor values. Table 4 provides the overall ranking of techniques, and the impact of complexity and the TBO factor.

The overall ranking of techniques is given in the first column of Table 4: with the parametrization that we have chosen the group consisting of ST, SA and HC outperforms the heuristics due to Graves, Heinrich and Schneeweiss, the best sequential technique (which is most often sequential WW) and the technique from Bookbinder and Koch. The excess cost ratio exhibited by specific techniques ranges from 7% for Graves' technique to 76% for the method of Bookbinder and Koch (the lot-for-lot solution was above 100% of excess cost, hence is not reported in the table). Inside the group of stochastic search techniques, the best option is simulated tempering with a (small) advantage of 0.3% over SA, and mostly for structures of low complexity. This result obtains because random temperature changes force the algorithm to wander across the whole space of potential candidates, and here global search seems to be more rewarding than localized exploration. Geometric annealing as in traditional SA comes next, and finally hill climbing with an excess cost of 6% on average. Regarding the importance of the factors defining the problems themselves, complexity (C) influences positively the relative performance of stochastic search as cost ratios increase from left to right in the table. Increasing the TBO factor tends to have a similar influence on the relative value of stochastic search, though less systematically (see G and BK).

Table 5
Execution times in CPU s, 200 × 24 test instances

Rule	CPU	Complexity (C)				TBO	
		0	0.25	0.5	0.75	(1,2)	(2,6)
ST	52.9	16.3	45.1	67.9	82.5	53.1	52.8
SA	61.4	14.1	53.5	79.9	98.1	73.3	49.6
HC	18.2	6.2	14.9	21.6	29.9	25.5	10.8

What these results tell us is that a sophisticated search rule together with a non-sophisticated annealing scheme tend to achieve the best trade-off. Regarding cooling, in a cost landscape that is as highly disordered as the one underlying the MLLS problem is, lowering the temperature is often responsible for entrapment in local minima. As there are many such equilibria and as neighbouring points (points that can be reached by application of the transition rule) can be associated with highly different costs (not to mention the very large number of infeasible candidates), the rugged cost landscape should be explored in a random manner, accepting almost any transition and simply covering the largest part of it. It is worth recalling that ST demands a specific stopping rule for temperature never converges (the system does not freeze by construction). What was implemented here is a fixed number of epochs equal to the average number of epochs demanded by geometric cooling.

In Table 5 we report execution times for ST, SA and HC in the 4 complexity configurations. As for time to convergence, on average it takes less than 1 s CPU for the specific heuristics to solve a 200 × 24 test case, when stochastic search demands an average 52.9 s (ST), 61.4 s (SA) and 18.2 s (HC). Complexity (C) has a very significant influence on the convergence time of stochastic search. Computation times change from simple to double from $C = 0$ to $C = 0.75$. So the issue is to know whether it is worth spending more than one minute running a stochastic search routine with regard to the improvement which can be reached. It seems that with figures running from 7% to 76% percent in this study, the answer could well be yes.

6. Conclusion

In this paper a large set of single point stochastic search algorithm was tested against a variety of small and large problems. The conclusion we came to is twofold. First, a general and un-careful implementation of stochastic search is, generally speaking, less cost efficient and demands larger amounts of computer time than problem specific techniques. Second, the benefits from stochastic search are dependent on the size of the problems that are being addressed. Large problems can be efficiently treated by carefully designed search routines (more efficiently than by problem specific techniques, with relative differences no less than 7%) but their computational demands can be much more important (up to several minutes for a 200×24 case). Furthermore, designing an efficient search routine is itself a complex task as problem specific features have to be embedded in the routine. Mostly these features are included in the transition rule that is used to generate new candidates. In the problem considered here, cumulative mutation affecting all ancestors turned out to be the most efficient design. The second central element in algorithm design is the annealing scheme. Given the ruggedness of the cost landscape underlying the MLLS program, randomly changing the temperature appears to be preferable to a monotonic scheme. This is due to the large number of local minima that characterize the search space.

Regarding the generality of the results presented here, it has to be said that although a fairly large number of algorithms have been tested, each of them consists in a combination of routines for which parameter values have to be specified. Rather than testing for the effect of different parametrizations on a given assemblage of routines, in this paper we have sought to identify efficient combinations of routines keeping the parametrization fixed. It is possible that browsing the space of parameter configurations (such as epoch length, the cooling parameter or the stopping condition) would have led to even better performance, at least for some combinations of routines.

How can search be further improved? Certainly more sophisticated transition rules can be proposed. A combination of mutation and cost modifications could be employed for instance, to account more precisely for inter-stage product interdependencies. This in a sense would go against the original spirit of stochastic search which aims to explore a given fitness landscape in a simple and robust manner. Beside the search rule, it is natural to consider alternative formulations of the annealing scheme. A useful check would consist in letting the system accept almost any candidate point, to see whether totally disregarding the effect of temperature could further improve performance. Another fruitful direction could be to employ a more sophisticated penalty scheme that would in a sense smoothen the cost landscape. By doing so it is possible that the system would be able to reach low cost regions surrounded by infeasible candidates.[3] The relevance of single point stochastic search when applied to the MLLS problem can itself be questioned. Whether there are more apt methods for such a complex cost landscape is an open question. One possible route is parallelizing search, which can for instance be implemented through a genetic algorithm (see for instance Dellaert and Jeunet [26]). An alternative way to preserve the single point feature of the approach while taking more information into account is to use tree annealing (Bilbro and Snyder [32]). In tree annealing there are successive partitions of the search space as search proceeds. A growing tree results, for which at each node a left/right decision is made probabilistically. The decision to move left or right is made on the basis of a measure of the number of times in the past that an acceptable (again in a probabilistic sense) point was found in the left and right sub-trees, respectively, of that node. Note that as search always begins at the root (the starting point), the entire tree is always available to be searched. However, the resulting partition tree is more likely to provide more resolution in the vicinity of minima.

Acknowledgements

We thank a referee of this journal for having provided very detailed and useful comments.

[3] This point was suggested by the referee.

References

[1] Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E. Equation of state calculations by fast computing machines. Journal of Chemical Physics 1953;21:1087–92.

[2] Kirkpatrick S, Gerlatt Jr. CD, Vecchi MP. Optimization by simulated annealing, science. Science 1983;220:671–80.

[3] Kuik R, Salomon S. Multi-level lot-sizing problem: evaluation of a simulated-annealing heuristic. European Journal of Operations Research 1990;45:25–37.

[4] Özdamar L, Barbarosoglu G. Hybrid heuristics for the multi-stage capacitated lot sizing and loading problem. Journal of the Operational Research Society 1999;50:810–25.

[5] Barbarosoglu G, Özdamar L. Analysis of solution space-dependent performance of simulated annealing: the case of the multi level capacited lot sizing problem. Computers and Operations Research 2000;27:895–903.

[6] Marinari E, Parisi G. Simulated tempering: a new Monte Carlo scheme. Europhysics Letters 1992;69:2292.

[7] Dueck G, Scheuer T. Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. Journal of Computational Physics 1990;90:161–75.

[8] Zangwill WI. Minimum concave cost flows in certain networks. Management Science 1968;14:429–50.

[9] Zangwill WI. A backlogging model and a multi-echelon model of a dynamic economic lot size production system—a network approach. Management Science 1969;15:506–27.

[10] Crowston WB, Wagner HM. Dynamic lot size models for multi-stage assembly systems. Management Science 1973;20:14–21.

[11] Steinberg E, Napier HA. Optimal multi-level lot sizing for requirements planning systems. Management Science 1980;26:1258–71.

[12] Afentakis P, Gavish B, Karmarkar U. Computationally efficient optimal solutions to the lot-sizing problem in multistage assembly systems. Management Science 1984;30:237–49.

[13] Afentakis P, Gavish B. Optimal lot-sizing algorithms for complex product structures. Operations Research 1986;34:237–49.

[14] Saydam C, Coleman BJ, McKnew MA. An efficient zero-one formulation of the multilevel lot-sizing problem. Decision Sciences 1991;22:280–95.

[15] Veral EA, LaForge RL. The performance of a simple incremental lot-sizing rule in a multilevel inventory environment. Decision Sciences 1985;16:57–72.

[16] Yelle LE. Materials requirements lot sizing: a multi-level approach. International Journal of Production Research 1979;17:223–32.

[17] Wagner HM, Whitin TM. Dynamic version of the economic lot size model. Management Science 1958;5:89–96.

[18] Blackburn JD, Millen RA. Improved heuristics for multi-stage requirements planning systems. Management Science 1982;28:44–56.

[19] Bookbinder JH, Koch LA. Production planning for mixed assembly/arborescent systems. Journal of Operations Management 1990;9:7–23.

[20] Graves SC. Multi-stage lot sizing: an iterative procedure. In: Schwarz LB, editor. Multi-level production/inventory control systems: theory and practice. Amsterdam: North-Holland; 1981.

[21] Heinrich CE, Schneeweiss C. Multi-stage lot-sizing for general production systems. In: Schneeweiss C, Silver E, Axsäter S, editors. Multi-stage production planning and inventory control. Berlin: Springer; 1986. p. 150–81.

[22] Lambrecht M, Van der Eecken J, Vanderveken H. Review of optimal and heuristic methods for a class of facilities in series dynamic lot-size problems. In: Schwarz LB, editor. Multi-level production/inventory control systems: theory and practice. Amsterdam: North-Holland; 1981. p. 69–93.

[23] Lambrecht M, Van der Eecken J, Vanderveken H. A comparative study of lot-sizing procedures for multi-stage assembly systems. OR-Spektrum 1983;5:33–43.

[24] Afentakis P. A parallel heuristic algorithm for lot sizing in multistage production system. IIE Transactions 1987;19: 34–42.

[25] Dellaert NP, Jeunet J, Jonard N. A genetic algorithm to solve the general multi-level lot-sizing problem with time-varying costs. International Journal of Production Economics 2000;68:241–57.

[26] Dellaert NP, Jeunet J. Solving large unconstrained multi level lot sizing problems using a hybrid genetic algorithm. International Journal of Production Research 2000;38:1083–99.

[27] Veinott Jr. AF. Minimum concave-cost solution of leontieff substitution models of multifacility inventory systems. Operations Research 1969;17:262–91.
[28] van Laarhoven PJM, Aarts EH. Simulated annealing: theory and applications. The Netherlands: Reidel Publishing; 1987.
[29] Gendreau M, Hertz A, Laporte G. A tabu search heuristic for the vehicle routing problem. Management Science 1994;40:1276–90.
[30] Silver EA, Meal HC. A heuristic for selecting lot size requirements for the case of a deterministic time-varying demand rate and discrete opportunities for replenishment. Production and Inventory Management 1973;14:64–74.
[31] Kimms A. Multi-level lot-sizing and scheduling methods for capacitated, dynamic, and deterministic models. Physica verlag series on production and logistics. Berlin: Springer; 1997.
[32] Bilbro G, Snyder W. Optimization of functions with many minima. IEEE Transactions on Systems, Man and Cybernetics 1991;21:840–9.