

# A cross entropy-Lagrangian hybrid algorithm for the multi-item capacitated lot-sizing problem with setup times

M. Caserta<sup>a,\*</sup>, E. Quiñonez Rico<sup>b</sup>

<sup>a</sup>*Institut für Wirtschaftsinformatik, Universität Hamburg, Von-Melle-Park 5, D - 20146 Hamburg, Germany*

<sup>b</sup>*Instituto Tecnológico de Monterrey, Calle del Puente, 222, Col. Ejidos de Huipulco, Del. Tlalpan, México DF 14380, Mexico*

Available online 11 October 2007

---

## Abstract

The aim of this article is to introduce a hybrid algorithm for the multi-item multi-period capacitated lot-sizing problem with setups. In this problem, demands over a finite planning horizon must be met, where several items compete for space with limited resources in each period, and a portion of these resources is used by setups. The proposed scheme considers a Lagrangian relaxation of the problem and applies a cross entropy-based metaheuristic to the uncapacitated version of the original problem. A thorough experimental plan has been designed and implemented to test the effectiveness and the robustness of the algorithm: first, drawing inspiration from the response surface methodology, we calibrate the algorithm by identifying the optimal parameters value for any given instance size. Next, we carry out experiments on large scale instances, collecting information about solution quality and computational time, and comparing these results with those offered by a global optimizer.

© 2007 Elsevier Ltd. All rights reserved.

*Keywords:* Metaheuristics; Production; Cross entropy; Lagrangian heuristics; Capacitated lot sizing

---

## 1. Introduction

The multi-item multi-period capacitated lot-sizing problem with setups (MCLS) is a well known optimization problem that finds a wide variety of applications in real-world problems. It is a well-known fact that MCLS is an  $\mathcal{NP}$ -hard problem [1–3]. For this reason, the problem is both challenging from the computational point of view as well as interesting from the real-world applicability perspective. In this paper, we propose a new metaheuristic-based algorithm for MCLS, along with some insight about how to fine-tune the proposed metaheuristic algorithm.

This section is organized as follows: first, the mathematical formulation is presented; next, a thorough literature review is offered, and, finally, an overview of the proposed scheme and of the overall paper is introduced.

---

\* Corresponding author. Tel.: +49 40 42838 3062; fax: +49 40 42838 5535.

E-mail addresses: [caserta@econ.uni-hamburg.de](mailto:caserta@econ.uni-hamburg.de) (M. Caserta), [quinonez@itesm.mx](mailto:quinonez@itesm.mx) (E.Q. Rico).

### 1.1. Problem formulation

A mixed-integer formulation of the problem is

$$\begin{aligned}
 \text{(MCLS): } \quad & \min \sum_{j=1}^P \sum_{t=1}^T (c_{jt}y_{jt} + f_{jt}x_{jt} + h_{jt}s_{jt}) \\
 \text{s.t.} \quad & \sum_{j=1}^P (a_{jt}y_{jt} + m_{jt}x_{jt}) \leq b_t, \quad t = 1, 2, \dots, T, \\
 & y_{jt} + s_{j,t-1} - s_{jt} = d_{jt}, \quad j = 1, 2, \dots, P, \quad t = 1, 2, \dots, T, \\
 & y_{jt} \leq Mx_{jt}, \quad j = 1, 2, \dots, P, \quad t = 1, 2, \dots, T, \\
 & y_{jt}, s_{jt} \geq 0, \quad x_{jt} \in \{0, 1\}, \quad j = 1, 2, \dots, P, \quad t = 1, 2, \dots, T.
 \end{aligned}$$

In this model,  $P$  and  $T$  stand for the number of items and the number of periods in the planning horizon, respectively. The model presents two groups of continuous variables, which account for the production and the inventory level of each item in each period ( $y_{jt}$  and  $s_{jt}$ , respectively). In addition, a group of binary variables  $x_{jt}$  is used to indicate whether any positive quantity of item  $j$  is produced during period  $t$ . Parameters  $c_{jt}$ ,  $f_{jt}$ , and  $h_{jt}$  stand for unitary production cost, setup cost, and inventory cost for item  $j$  during period  $t$ , respectively. Production capacity during period  $t$ , e.g., available production time, is indicated by  $b_t$ . Finally,  $a_{jt}$  and  $m_{jt}$  represent the unitary production time and the setup time for product  $j$  during period  $t$ , respectively. The first set of constraints accounts for the production capacity limit over the periods and ensures that the overall production does not require more than the available capacity. The second set of constraints, inventory balance constraints, ensures that, for each period and each item, demands are satisfied. Finally, the third set of constraints ensures that appropriate setup costs are paid if some positive production is taken. An illustrative example, along with a graphical representation of the lot sizing problem over a number of time periods is presented in Nemhauser and Wolsey [3].

### 1.2. Literature review

Due to its vast industrial applicability, researchers have devoted special attention to MCLS (see [4–7]). The problem is complicated by the existence of setup costs as well as inventory costs. For this reason, a trade off exists between producing an item at every period, and, consequently, paying less inventory but more setup, or producing more sparingly, hence paying less setup but more inventory. Since the multi-item capacitated lot-sizing problem with setups is very difficult to solve to optimality, many researchers have tried to tackle the problem by working on relaxations of the same. A good description of some well studied relaxations of MCLS is provided by Miller et al. [8].

Two main approaches can be identified in the currently available literature. On the one hand, a number of authors have tried to tackle the problem from a general perspective, either defining tighter formulations or via valid inequalities generation. Generally speaking, when the linear programming relaxation of an MIP problem is tight, the optimal objective value of such problem is close to the optimal value of the integer problem. Consequently, a branch and bound-based algorithm could efficiently solve the integer problem. These approaches, typically proposed in the 1980s and early 1990s are inherently flexible, and thus can easily be adapted to solve a wide range of MIP problems, including the MCLS problem. On the other side of the spectrum we find specialized algorithms, mainly based upon Lagrangean relaxation along with an ad hoc heuristic. While these algorithms typically require a good deal of modifications even for small changes in the model, they are more efficient than any general approach and can be used to solve larger instances in a shorter computational time. Recent contributions have been focused on such avenue of research, proposing ad hoc algorithms for MCLS able to efficiently solve instances with tens of thousand of variables.

As an example of what can be seen as a ‘general approach’, Eppen and Martin [5] formulate the capacitated lot-sizing problem as a shortest route problem and solve such network problem using a standard “off-the-shelf” solver. The general idea of the paper is to exploit the structure of the subproblem (uncapacitated lot sizing) to define a tighter

formulation. It is worth noting that the linear programming relaxation of the model resulting from variable redefinition produces bounds equal to those of the Lagrangean relaxation. They solve MCLS problems with up to 200 products and 10 periods in a reasonable amount of computational time.

Similarly, Pochet and Wolsey [6] present different variants of the lot-sizing problem. Their approach is based upon the generation of valid inequalities that strengthen the linear programming formulation. In turn, good bounds provided by the LP relaxation foster the effectiveness of a branch and bound-based approach for the general lot-sizing problem. The authors present results on instances with up to five items and 12 periods, which are solved to optimality in a short amount of computational time.

Some other studies, concerning the construction of valid inequalities, have been developed by Miller et al. [8]. This author provides studies on valid inequalities for different relaxations of MCLS, namely the single-item multi-period and the multi-item single-period lot-sizing problems. With respect to the single-item problem, further indications are provided in [6,9,10].

In a different study, Miller et al. [11] present a special case of the multi-item single-period capacitated lot sizing problem with setups, where setup times and demands are constant for all items. For this special case, the authors provide a polynomial time algorithm. They prove that the inequalities identified in Miller et al. [12] suffice to solve such problem by linear programming in polynomial time.

Another study on valid inequalities is the one presented in Loparic et al. [13]. They derive facet-defining inequalities for the dynamic knapsack set as well as for the single-item capacitated lot-sizing problem. Computational results offered are quite limited, even though the major point made by the authors is that dynamic knapsack inequalities and their generalizations provide strong valid inequalities for the lot-sizing problem.

Along the same line, Magnanti and Sastry [14] present a study of different scheduling problems. They first derive valid inequalities for the single-item capacitated lot-sizing problem and, subsequently, apply their findings to the multi-item version. They offer an interesting comparison of different strategies, providing a measure of the tightness of the LP model with four families of valid inequalities. Computational results with up to five products and 30 periods are presented. For all these instances, at least one of the proposed formulations with valid inequalities closes the gap between the LP relaxation and the MIP problem.

Considering specialized approaches to MCLS, the seminal work of Trigeiro et al. [2] illustrate how Lagrangean relaxation coupled with subgradient optimization can be used to solve small-medium size instances of MCLS. For each Lagrangean vector, they apply a dynamic programming algorithm to independently solve to optimality a series of uncapacitated lot-sizing problems, one for each item. If the Lagrangean solution is not feasible, a smoothing procedure is applied in order to restore feasibility whenever possible. They present results on random generated instances with up to 24 items and 30 periods. In order to evaluate the goodness of the algorithm, they perform a detailed analysis of the solution gap, computed as distance between the integer solution and the best lower bound. Such gap averaged less than 4%, when computed on instances with a number of items between 6 and 24 and a number of periods between 15 and 30.

Diaby et al. [15] present an ad hoc Lagrangean-based algorithm for large scale MCLS problems that achieves excellent results. They apply Lagrangean relaxation with subgradient optimization to get good lower bounds and then use a branch and bound scheme to obtain the optimal solution to the problem. They solve instances with thousands of items within a 1% of their optimal solution.

Tempelmeier and Derstoff [16] tackle MCLS in a similar way, decomposing the original problem in a set of uncapacitated problems that are solved to optimality. They solve MCLS as the core planning step of the broader Material Requirements Planning system and consider the multi-level version of the problem, where one item is composed of a number of subassembly operations.

In a different study, Katok et al. [17] propose a two-phase algorithm for MCLS. The first phase is aimed at finding a good feasible solution via a Coefficient Modification Heuristic. Next, an LP problem with added constraints is solved to improve the first stage solution. They solve problems with up to 5000 variables and obtain solutions which average 25% better than solutions obtained with a commercial solver in a fixed amount of computational time.

The approach presented by Hindi et al. [18] is interesting for its use of a metaheuristic scheme within a Lagrangean relaxation scheme solved via subgradient optimization. Furthermore, these authors provide an interesting explanation of why the MCLS problem is much more difficult than its counterpart without setup times. The algorithm uses a smoothing heuristic embedded into a Lagrangean relaxation scheme coupled with a variable neighborhood local search scheme. The solution found by the smoothing scheme is further improved by taking it as starting point for a variable

neighborhood search algorithm. The proposed heuristic is effectively used to tackle problems with up to 24 items and 30 periods.

In a different study, Sambasivan and Schmidt [19] apply the Lagrangean relaxation technique to the multi-level version of the MCLS problem, which is, the lot-sizing problem with multiple plants. The relaxed problem is solved using an efficient heuristic developed by the authors. Random generated instances are then solved to show the effectiveness of the proposed algorithm.

Finally, we present two metaheuristic-based schemes for the MCLS since a more direct comparison with our algorithm can be performed. Gopalakrishnan et al. [20] present a Tabu Search heuristic for the capacitated lot-sizing problem with the variant of setup carryover, in which carrying items from one period to the next requires the payment of a setup cost. They define a number of dynamic moves and allow the partial exploration of the infeasible space. Finally, they introduce a lower bounding heuristic to improve the quality of the tabu search solution. The algorithm was tested on 751 instance problems from Trigeiro et al. [2], where the setup carryover is set to zero.

### 1.3. Contributions

The major contribution of this paper concerns the development of a hybrid scheme that reduces MCLS to a series of uncapacitated lot-sizing problems (ULS), each one of them effectively solved using the meritorious cross entropy (CE) paradigm. This work is based upon some of the findings of Caserta et al. [21], in which the integer knapsack problem with setup was effectively solved using a CE-based scheme. The integer knapsack problem with setup is viewed as a particular case of the multi-item capacitated lot-sizing problem with setup times, as presented in Miller et al. [8] and, consequently, the CE scheme presented in this paper draws inspiration from the scheme proposed for the knapsack problem. In this paper, we will show that, after dualizing the capacity constraints, MCLS reduces to a set of  $P$  uncapacitated lot-sizing problems that can be separately solved using the CE framework. Hence, MCLS is solved by progressively determining “good” Lagrangean multipliers so that the joint solution of the set of ULS problems is feasible to MCLS. Furthermore, the paper illustrates how statistical techniques, such as the response surface methodology, can be used to perform a good calibration of the metaheuristic algorithm. This analysis will help to identify the relation between instances’ characteristics and problem difficulty.

This paper has the following structure: Section 2 illustrates how the CE paradigm can be adapted to solve uncapacitated lot-sizing problems. Next, Section 3 presents the overall dual scheme used to tackle MCLS, which is, it shows how the CE scheme is embedded into the Lagrangean Optimization phase. In Section 4 we present a greedy scheme used to project a Lagrangean solution, infeasible to MCLS, into the feasible space of the problem. This scheme is used to derive a feasible solution at every Lagrangean iteration. Section 5 is devoted to the presentation of the experimental plan, designed with a two-fold objective in mind: on the one hand, it will be shown how the Cross Entropy-based algorithm is calibrated while, on the other hand, we will prove the effectiveness of the proposed scheme on very large scale instances of MCLS. Finally, Section 6 presents some concluding remarks and possible future works.

## 2. CE algorithm for uncapacitated lot-sizing problem

The main idea of CE [22] is related to the design of an effective learning mechanism throughout the search process. It associates an estimation problem to the original combinatorial optimization problem, called the associated stochastic problem, characterized by a density function  $\phi$ . The stochastic problem is solved by identifying the optimal importance sampling (IS) density  $\phi^*$ , which is the one that minimizes the Kullback–Leibler distance with respect to the original density  $\phi$ . This distance is also called the cross-entropy between  $\phi$  and  $\phi^*$ . The minimization of the cross-entropy leads to the definition of “optimal” updating rules for the reference parameter of the density functions and, consequently, to the generation of improved feasible vectors. The method terminates when convergence to a point in the feasible region is achieved.

In this paper we present a brief introduction about how CE is used to solve the single-item uncapacitated lot-sizing problem (see also Caserta et al. [21]).

Let us consider the general 0–1 integer minimization problem

$$(P): \quad z^* = \min_{\mathbf{x} \in \mathcal{X}} S(\mathbf{x}),$$

where  $\mathcal{X} \subset \mathbb{B}^n$  represents the feasible region. The CE method associates a stochastic estimation problem to (P) aimed at estimating

$$\mathbb{P}(S(\mathbf{x}) \leq z).$$

With this aim, let us define a family of density functions  $\phi$  on  $\mathcal{X}$ , parametrized by a vector  $\mathbf{u} \in [0, 1]^n$ . Since we want to generate binary vectors to identify the production periods of each item, let us assume that we use a family of Bernoulli density functions under the following pdf:

$$\phi(\mathbf{x}, \mathbf{u}) = \prod_{j=1}^n (u_j)^{x_j} (1 - u_j)^{1-x_j}. \tag{1}$$

In the case of the uncapacitated lot-sizing problem, a binary vector  $\mathbf{X}_k$  associated to a given item suggests in which periods should the production be enforced, by taking as production periods those periods for which  $X_{kj} = 1$ .

The stochastic estimation problem associated to the original combinatorial optimization problem is

$$(EP): \quad \mathbb{P}_{\mathbf{u}}(S(\mathbf{X}) \leq z) = \sum_{\mathbf{x} \in \mathcal{X}} I_{\{S(\mathbf{x}) \leq z\}} \phi(\mathbf{x}, \mathbf{u}),$$

where  $I_{\{S(\mathbf{x}) \leq z\}}$  is the indicator function, whose value is 1 if  $S(\mathbf{x}) \leq z$  and 0 otherwise, and  $\mathbb{P}_{\mathbf{u}}$  is the probability measure that a random state  $\mathbf{X}$ , drawn under  $\phi(\cdot, \mathbf{u})$ , has performance function value less than or equal to a given threshold value  $z$ .

We want to estimate  $l = \mathbb{P}_{\mathbf{u}}(S(\mathbf{x}) \leq z)$  for  $z = z^*$ . When the size of  $\mathcal{X}$  is large enough, simulation is an acceptable approach to find  $\mathbf{x}^*$  such that  $z^* = S(\mathbf{x}^*) \leq S(\mathbf{x})$  for all  $\mathbf{x} \in \mathcal{X}$ . We could estimate  $l$  via Monte Carlo simulation by drawing a random sample  $\mathbf{X}_1, \dots, \mathbf{X}_k, \dots, \mathbf{X}_N$  from  $\phi(\mathbf{x}, \mathbf{u})$  with

$$\frac{1}{N} \sum_{k=1}^N I_{S(\mathbf{X}_k) \leq z}.$$

However, the event  $\mathbf{X}_k = \mathbf{x}^*$  is a rare event, with probability  $1/|\mathcal{X}|$ . Hence, the size of the sample required to estimate  $l$  might be extremely large. Alternatively, it is possible to estimate  $l$  via importance sampling (IS). Let us take a random sample  $\mathbf{X}_1, \dots, \mathbf{X}_k, \dots, \mathbf{X}_N$  from a different density function  $\phi(\mathbf{x}, \mathbf{p})$ . In this case, an estimator of  $l$ , indicated by  $\hat{l}$ , is given by the likelihood ratio estimator

$$\hat{l} = \frac{1}{N} \sum_{k=1}^N I_{\{S(\mathbf{X}_k) \leq z\}} \frac{\phi(\mathbf{x}, \mathbf{u})}{\phi(\mathbf{x}, \mathbf{p})},$$

where  $\phi(\mathbf{x}, \mathbf{p})$  constitutes a change of measure and is chosen such that the CE between  $\phi(\mathbf{x}, \mathbf{p})$  and  $\phi(\mathbf{x}, \mathbf{u})$  is minimal. The objective here is to determine the optimal reference parameter associated with  $\mathbb{P}_{\mathbf{p}}(S(\mathbf{X}) \leq z)$ . Parameter  $\mathbf{p}$  can be estimated by solving the problem

$$\hat{\mathbf{p}} = \operatorname{argmax}_{\mathbf{p}} \frac{1}{N} \sum_{k=1}^N I_{\{S(\mathbf{X}_k) \leq z\}} \ln \phi(\mathbf{X}_k, \mathbf{p}),$$

where  $\mathbf{X}_k$  are generated under pdf  $\phi(\cdot, \mathbf{p})$ . Since  $\phi(\cdot, \mathbf{p})$  is given by Eq. (1), and since each component  $X_{kj}$  only takes values 0 or 1, finding  $\hat{\mathbf{p}}$  under Eq. (2) corresponds to solving

$$\frac{\partial}{\partial p_j} \sum_{k=1}^N I_{\{S(\mathbf{X}_k) \leq z\}} \ln \phi(\mathbf{X}_k, \mathbf{p}) = 0,$$

which gives the optimal updating rule:

$$\hat{p}_j = \frac{\sum_{k=1}^N I_{\{S(\mathbf{X}_k) \leq z\}} x_{kj}}{\sum_{k=1}^N I_{\{S(\mathbf{X}_k) \leq z\}}}, \quad j = 1, \dots, n, \tag{2}$$

where  $\mathbf{X}_k$  indicates a binary vector and  $x_{kj}$  is the  $j$ th component of the  $k$ th point of the CE population.

If we establish an arbitrary initial value for  $\mathbf{p}^0$ , rule (2) can be iteratively used with the aim of generating a sequence of increasing threshold values  $z^0, z^1, \dots$ , converging either to the global optimum  $z^*$  or to a value close to it. At each iteration  $t$ , the new value of  $z^t$  is used to generate a better vector  $\mathbf{p}^{t+1}$ . The new vector  $\mathbf{p}^{t+1}$  is, in turn, used to draw a new, hopefully better, sample population, which will lead to a better value of  $z^{t+1}$ . The process stops when either we reach the global optimum value  $z^*$  or the vector  $\mathbf{p}$  converges to a vector in  $\mathcal{X}$ , which implies that any random state drawn under  $\phi(\cdot, \mathbf{p})$  will converge to the same solution in  $\mathcal{X}$ .

Let us consider now how CE can be adapted to solve the uncapacitated lot-sizing problem. We only present the case of a single-item uncapacitated lot-sizing problem. Results can easily be extended to the multi-item case, by considering parallel machines. A mixed-integer formulation of the problem is

$$\begin{aligned}
 \text{(ULS): } \quad & \min \quad \sum_{t=1}^T (c_t y_t + f_t x_t + h_t s_t) \\
 \text{s.t.} \quad & y_t + s_{t-1} - s_t = d_t, \quad t = 1, 2, \dots, T \\
 & y_t \leq M x_t, \quad t = 1, 2, \dots, T \\
 & s \in \mathbb{R}_+^{T+1}, \quad y \in \mathbb{R}_+^T, \quad x \in \mathbb{B}^T.
 \end{aligned}$$

The model is basically the same as MCLS, with the exception that ULS deals with a single item, rather than a set of  $P$  items. Obviously, the first set of constraints of MCLS is not present in ULS, since there are no capacity constraints.

The following notation is used to describe a dynamic programming solution to ULS. Let  $z_t(s_{t-1})$  be the optimal solution to ULS considering only periods  $t, t + 1, \dots, T$  and assuming that inventory at the beginning of period  $t$  is  $s_{t-1}$ . The following dynamic recursion, presented in [3], provides an optimal solution to ULS:

$$z_t(s_{t-1}) = \begin{cases} \min_{\substack{x_k \in \{0,1\} \\ 0 \leq y_k \leq M x_k}} \{c_t y_t + f_t x_t + h_t s_t\}, & t = T, \\ \min_{\substack{x_k \in \{0,1\} \\ 0 \leq y_k \leq M x_k}} \{c_t y_t + f_t x_t + h_t s_t + z_{t+1}(s_t)\}, & t = 1, \dots, T - 1, \end{cases} \tag{3}$$

where  $s_t = s_{t-1} + y_t - d_t, t = 1, \dots, T$ .

The following theorem, presented in [3], provides strong conditions on the values of  $s_{t-1}$  and  $y_t$  and is used to solve ULS via (3).

**Theorem 1.** *There is an optimal solution  $(y_t^*, x_t^*, s_{t-1}^*)$  to ULS, such that:*

- (i) for all  $t = 1, 2, \dots, T, s_{t-1} y_t = 0$ ;
- (ii) if  $y_t > 0$ , then  $y_t = \sum_{k=t}^p d_k$  for some  $t \leq p \leq T$ ;
- (iii) if  $s_{t-1} > 0$ , then  $s_{t-1} = \sum_{k=t}^q d_k$  for some  $t \leq q \leq T$ .

Owing to Theorem 1, problem ULS reduces to the identification of production periods. If we knew for which periods  $t$  the variable  $x_t$  should be set to 1, the associate solution of ULS could be easily computed applying (ii) and (iii) of Theorem 1. Consequently, a solution to ULS can be uniquely defined by a binary vector  $\mathbf{X} \in \mathbb{B}^T$ , where each bit  $X[t]$  is set to 1 if the item is to be produced in period  $t$ , and to 0 otherwise. In line with this reasoning, we define a CE-based scheme aimed at “guessing” which periods should be used to produce a given item. The scheme generates a population of binary vectors under a Bernoulli distribution of parameter  $\mathbf{p}$ , evaluates each binary solution in terms of objective function value of ULS using Theorem 1, and updates the parameter vector of the Bernoulli distribution with the aim of generating a better population in the next iteration.

It is worth noting that, since we assume  $d_1 > 0$  and  $s_0 = 0$ , the first bit of each binary vector must be set to 1. For this reason, we set  $y_1 \geq d_1, x_1 = 1, s_1 = y_1 - d_1$ .

Given a binary vector  $\mathbf{X} \in \mathbb{B}^T$ , where  $X[t]$  indicates the “ $t$ th” bit of the vector, the objective function value of  $\mathbf{X}$  is evaluated as follows: assuming that  $\{|t = 1, \dots, T : X[t] = 1\} = p$ , define  $l_i$  for  $i = 1, 2, \dots, p$  the set of positions

of bits set to one in  $X$ , such that  $l_1 < l_2 < \dots < l_p$ ; additionally, define  $l_{p+1} = T + 1$ . A feasible solution to ULS is given by

$$\hat{y}_t = \begin{cases} \sum_{k=l_i}^{l_{i+1}-1} d_k & \text{if } t = l_i \text{ for some } i \in \{1, 2, \dots, p\}, \\ 0 & \text{otherwise,} \end{cases}$$

$$\hat{s}_{t-1} = \begin{cases} \sum_{k=t}^{l_{i+1}-1} d_k & \text{if } l_i + 1 \leq t \leq l_{i+1} - 1 \text{ for some } i \in \{1, 2, \dots, p\}, \\ 0 & \text{otherwise.} \end{cases}$$

Finally, since the value of  $x_t$  is determined by the value of  $y_t$ , with these formulae we may compute the value of the objective function evaluated at the current solution, which is

$$S(\mathbf{X}) = \sum_{t=1}^T (c_t \hat{y}_t + f_t x_t + h_t \hat{s}_t) \quad \text{for } \mathbf{X} \in \mathbb{B}^T.$$

In the following, we provide the CE procedure for ULS as pseudo-code. As illustrated in steps 6 and 7, the algorithm terminates when either the probability vector  $\mathbf{p}$  converges to a binary vector or when a pre-fixed maximum number of iterations (`max_iter`) is reached.

**Algorithm 1.** CE-ULS

**Require:**  $\rho, N, \text{max\_iter}$

**Ensure:** feasible solution to ULS

- 1: generate initial Bernoulli probabilities  $\mathbf{p}^0 \in (0, 1)$ . Set  $t := 0$
- 2: draw a sample population  $\Omega^t = \{\mathbf{X}_1, \dots, \mathbf{X}_N\} \sim \text{Ber}(\mathbf{p}^t)$
- 3: compute  $S(\mathbf{X}_i)$  for each  $\mathbf{X}_i \in \Omega^t$  {solve (3) via Theorem 1}
- 4: sort  $\Omega^t$  in ascending order w.r.t.  $S(\mathbf{X}_i)$
- 5: compute  $\mathbf{p}^{t+1}$  as: {use updating rule (2)}

$$p_j^{t+1} = \frac{\sum_{i=0}^{\lceil \rho N \rceil} X_i[t]}{\lceil \rho N \rceil}, \quad j = 1, \dots, T$$

- 6: **if**  $\mathbf{p}^{t+1} \in \mathbb{B}^T \vee t = \text{max\_iter}$  **then**
- 7: STOP. {termination criteria}
- 8: **else**
- 9:  $t \leftarrow t + 1$  and go back to step 2
- 10: **end if**

**Remark 1.** As presented in Rubinstein [23], it is possible to compute the empirical computational complexity of algorithm CE\_ULS, which can be defined as

$$k = \tau_T(NG_T + NZ_T + S_N),$$

where  $\tau_T$  is the total number of iterations needed before the algorithm stops,  $N$  is the sample size,  $G_T = \mathcal{O}(T)$  is the cost of generating a binary vector under a Bernoulli distribution,  $Z_T = \mathcal{O}(T^2)$  indicates the cost of computing the objective function value of a binary vector  $\mathbf{X}_i$ , and  $S_N = \mathcal{O}(N \log N)$  is the cost of sorting  $N$  binary vectors.

We empirically found that  $\tau_t = \mathcal{O}(\log T)$ . For this reason, the theoretical complexity of algorithm CE\_ULS is  $\mathcal{O}(NT^2 \log T)$ , the empirical computational complexity is found to be  $\mathcal{O}(N \log T)$ , where the sample size  $N$  depends on the number of items and the number of periods, as illustrated in Section 5.1.

Table 1  
Comparison

No. periods	Time DP <sup>a</sup>	Time CE <sup>a</sup>	$N$	$\gamma^{\text{CE}}$
100	0.06	0.06	214	0
200	0.10	0.11	247	0
300	0.14	0.12	266	0
400	0.20	0.18	279	0
500	0.42	0.26	289	0
1000	3.62	0.83	322	0
1500	12.9	1.32	341	0.001
2000	30.7	3.02	354	0.002
5000	288.3	6.83	397	0.147

<sup>a</sup>Time measures in CPU seconds.

**Remark 2.** It is worth noting that the CE-based algorithm does not guarantee that the optimal solution to ULS will be found, especially when  $T$  is very large. However, we prefer to use the heuristic scheme, rather than exact algorithms based upon the Wagner–Within algorithm, for its advantages in terms of computational time. We conducted computational experiments with very large scale instances of the problem and we observed that the running time of the CE-based algorithm was shorter than that of a Wagner–Within based algorithm, such as, for example, the one of Aggarwal and Park [24].

We include here a summary of some empirical results when comparing the CE-based algorithm with the best algorithm available for the ULS problem. Wolsey [25], Karimi et al. [26], and Brahimy et al. [27] mention that exact algorithms with complexity  $\mathcal{O}(T \log T)$  were independently found by Wagelmans et al. [28], Aggarwal and Park [24], Federgruen and Tzur [29], and Van Hoesel et al. [30].

We implemented the algorithm proposed by Aggarwal and Park [24] and run experiments in order to compare the performance of such algorithm with the proposed CE-based method. We run the two algorithms on large scale ULS instances, with  $T$  varying from 100 to 5000. All the experiments have been carried out on an Intel Pentium 4 machine with 256 MB of RAM memory running Linux. Software code was written in C++ and compiled using the GNU g++ compiler. Results are summarized in Table 1.

In Table 1, the first column provides the instance size, which is, the number of periods; the second column gives the CPU time needed for the Aggarwal and Park's algorithm to find the optimal solution; the third column indicates the CPU time used by the CE-based algorithm to converge; column four gives the CE population size  $N$ , whose value is determined via statistical analysis (similar to what presented in Section 5.1 of the paper); finally, column five gives the relative error of the CE-based algorithm, computed as

$$\gamma^{\text{CE}} = \frac{z^{\text{CE}} - z^{\text{DP}}}{z^{\text{DP}}}.$$

From the table, we can compute the empirical computational complexity, as presented in Rubinstein [23], which is  $\mathcal{O}(N \log T)$  ( $p$ -value of 0.000027 and *Adjusted R*-value of 0.98). However, since  $N = \mathcal{O}(\log T)$ , we get that the average computational complexity of the CE-based algorithm is  $\mathcal{O}((\log T)^2)$ .

Observing Table 1, it is possible to see that the CE-based algorithm is not able to find the optimal solution to ULS when  $T$  is very large. However, we prefer to use the heuristic scheme, rather than the exact algorithm, for its advantages in terms of computational time. It is worth remembering that the original problem to be tackled is the capacitated multi-item multi-period lot-sizing problem (MCLS) and that each ULS problem is obtained as a relaxation of MCLS. Since there is no guarantee that the union of optimal solutions to the ULS problems will give a feasible solution to MCLS (especially in the first iterations of the subgradient optimization method, when the Lagrangean multipliers  $\mathbf{u}$  are far from being optimal), we prefer to give up something in terms of solution quality of ULS, accepting a quasi-optimal solution of the CE-based scheme and gaining in terms of running time.

### 3. Lagrangean-based metaheuristic algorithm for MCLS

Let us consider again the problem MCLS. Dualizing capacity constraints with a set of Lagrangean multipliers  $u_t \geq 0, t = 1, \dots, T$ , gives rise to a new problem with the following objective function:

$$\min \sum_{j=1}^P \sum_{t=1}^T (c_{jt}y_{jt} + f_{jt}x_{jt} + h_{jt}s_{jt}) + \sum_{j=1}^P \sum_{t=1}^T u_t ((a_{jt}y_{jt} + m_{jt}x_{jt}) - b_t).$$

Rearranging terms, we obtain

$$z(\mathbf{u}) = \sum_{j=1}^P \sum_{t=1}^T ((c_{jt} + u_t a_{jt})y_{jt} + (f_{jt} + u_t m_{jt})x_{jt} + h_{jt}s_{jt}) - \sum_{t=1}^T u_t b_t. \quad (4)$$

Notice that, for a given vector  $\mathbf{u} \in \mathbb{R}_+^T$ , the last term is constant and, consequently, can be dropped. If we set  $\hat{c}_{jt} = c_{jt} + u_t a_{jt}$  and  $\hat{f}_{jt} = f_{jt} + u_t m_{jt}$ , the Lagrangean relaxation of MCLS is

$$\begin{aligned} \text{(MCLS}(\mathbf{u})) \quad \min \quad & \sum_{j=1}^P \sum_{t=1}^T (\hat{c}_{jt}y_{jt} + \hat{f}_{jt}x_{jt} + h_{jt}s_{jt}) \\ \text{s.t.} \quad & y_{jt} + s_{j,t-1} - s_{jt} = d_{jt}, \quad j = 1, \dots, P, \quad t = 1, \dots, T, \\ & y_{jt} \leq Mx_{jt}, \quad j = 1, \dots, P, \quad t = 1, \dots, T, \\ & y_{jt}, s_{jt} \geq 0, x_{jt} \in \{0, 1\}, \quad j = 1, \dots, P, \quad t = 1, \dots, T. \end{aligned}$$

It can be observed that problem MCLS( $\mathbf{u}$ ) is separable into  $P$  different ULS problems, one for each type of item. Let us define ULS $_j(\mathbf{u})$  as the uncapacitated lot-sizing problem for item  $j$ , with  $j = 1, \dots, P$ . Such a problem has the same formulation as ULS of Section 2. Each ULS $_j(\mathbf{u})$  can be heuristically solved using the CE algorithm for ULS presented in Section 2.

Since MCLS( $\mathbf{u}$ ) is a relaxation of MCLS, a solution  $(\hat{\mathbf{y}}(\mathbf{u}), \hat{\mathbf{x}}(\mathbf{u}), \hat{\mathbf{s}}(\mathbf{u}))$  feasible to MCLS( $\mathbf{u}$ ), provides a lower bound, in terms of objective function value, for MCLS.<sup>1</sup> In order to simplify the notation, let us indicate with  $(\hat{\mathbf{y}}, \hat{\mathbf{x}}, \hat{\mathbf{s}})$  a solution to MCLS( $\mathbf{u}$ ) and with  $(\hat{\mathbf{y}}_j, \hat{\mathbf{x}}_j, \hat{\mathbf{s}}_j)$  a solution to ULS $_j(\mathbf{u})$ ,  $j = 1, \dots, P$ , found using the CE algorithm of Section 2, where solutions depend on the current value of  $\mathbf{u}$ . Once all the ULS $_j(\mathbf{u})$  have been solved, a solution to MCLS( $\mathbf{u}$ ), indicated by  $(\hat{\mathbf{y}}, \hat{\mathbf{x}}, \hat{\mathbf{s}})$ , is obtained as  $\hat{\mathbf{y}} = (\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \dots, \hat{\mathbf{y}}_P)$ ,  $\hat{\mathbf{x}} = (\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \dots, \hat{\mathbf{x}}_P)$ , and  $\hat{\mathbf{s}} = (\hat{\mathbf{s}}_1, \hat{\mathbf{s}}_2, \dots, \hat{\mathbf{s}}_P)$ . In order to find the tightest lower bound for MCLS, we need to solve the Lagrangean dual problem, defined as

$$L_D(\mathbf{u}) = \max_{\mathbf{u} \in \mathbb{R}_+^T} \left\{ \text{MCLS}(\mathbf{u}) \right\}_{\mathbf{y}, \mathbf{x}, \mathbf{s}}.$$

Problem  $L_D(\mathbf{u})$  is a linear problem in  $\mathbf{u}$  that is solved to approximation via the subgradient optimization technique, by generating a sequence of  $\{\mathbf{u}^k\}$  values, with  $k = 0, 1, 2, \dots$ , until a desired degree of convergence is achieved. Since we cannot guarantee that  $\mathbf{u}^*$  such that the objective function values of MCLS and MCLS( $\mathbf{u}^*$ ) are identical will be found, the aim here is to find near-optimal Lagrangean multipliers, so that the solution of MCLS( $\mathbf{u}$ ) is feasible, or almost feasible, to the original problem, MCLS. It is a well known fact that, within the context of Lagrangean relaxation theory, good feasible solutions to the original problem can frequently be obtained by perturbing nearly feasible solutions to the Lagrangean problem [31]. For this reason, while looking for near-optimal multipliers, at each iteration of the subgradient algorithm we apply a greedy scheme aimed at quickly modify the (infeasible) Lagrangean solution  $(\hat{\mathbf{y}}, \hat{\mathbf{x}}, \hat{\mathbf{s}})$  to reach a solution feasible to MCLS. More details about the greedy scheme are presented in Section 4.

<sup>1</sup> It is worth noting that a solution  $(\hat{\mathbf{y}}(\mathbf{u}), \hat{\mathbf{x}}(\mathbf{u}), \hat{\mathbf{s}}(\mathbf{u}))$  feasible to MCLS( $\mathbf{u}$ ) is guaranteed to be a lower bound for MCLS only if such a solution is optimal to MCLS( $\mathbf{u}$ ). Since we use a heuristic scheme to solve ULS( $\mathbf{u}$ ), we cannot guarantee that a solution obtained using the cross entropy scheme be optimal. Thus, the heuristic solution can only be used to compute an “approximate” lower bound. Such lower bound is computed using Eq. (4).

For the subgradient optimization algorithm, we use the formula of Held and Karp [31]:

$$u_t^{k+1} = \max \left\{ u_t^k + \lambda \frac{\sigma_t(u^k)}{\|\sigma(u^k)\|^2}, 0 \right\}, \quad t = 1, \dots, T, \quad (5)$$

where  $\lambda$  is the step size parameter, and  $\sigma_t(\mathbf{y}^k, \mathbf{x}^k, \mathbf{s}^k) = a_{jt}y_{jt} + m_{jt}x_{jt} - b_t$  is the component  $t$  of the subgradient, with  $t = 1, \dots, T$ .

The Lagrangean multipliers vector  $\mathbf{u}^0$  is initialized as

$$u_t^0 = \frac{\sum_{j=1}^P a_{jt}d_{jt} + m_{jt}}{b_t}, \quad t = 1, \dots, T, \quad (6)$$

in such a way that  $u_t^0$  is somehow proportional to the “pressure” for using the whole amount of capacity in a period in order to satisfy demands in that period. Step size  $\lambda$  is updated after every  $p = 20$  iterations, utilizing the best and worst lower bounds information obtained during the last  $p$  iterations. The steps of the Lagrangean Optimization phase are summarized in Procedure *Lagrangean Optimization*. In the following, we indicate with  $z(\mathbf{u})$  the objective function value of a solution  $(\hat{\mathbf{y}}, \hat{\mathbf{x}}, \hat{\mathbf{s}})$  of the Lagrangean problem MCLS( $\mathbf{u}$ ), for a given value of multipliers  $\mathbf{u}$ . In addition, let  $LB_{\text{best}}$  and  $LB_{\text{worst}}$  be the best and the worst lower bound obtained in the last  $p$  iterations.

#### Algorithm 2. Lagrangean Optimization

**Require:**  $LB, \mathbf{u}^0$

**Ensure:**  $LB, \mathbf{y}^*, \mathbf{x}^*, \mathbf{s}^*$

```

k ← 0           {Lagrangean iteration counter}
λ = 0.1
while lower bound termination tolerance is not met do
  if (k mod 20) = 0 then
    if  $LB_{\text{best}} - LB_{\text{worst}} > 0.01LB_{\text{best}}$  then
      λ ← 0.5λ
    else
      λ ← 1.5λ
    end if
     $LB_{\text{best}} = LB_{\text{worst}} = z(\mathbf{u}^k)$            {set best and worst LB}
  end if           {new step size available}
  k ← k + 1       {increase Lagrangean iteration counter}
  update  $\mathbf{u}^k$  via (5)   {perform kth Lagrangean iteration}
  solve MCLS( $\mathbf{u}^k$ ) via CE-ULS           {See Section 2}
  if  $z(\mathbf{u}^k) > LB$  then
     $LB \leftarrow z(\mathbf{u}^k)$            {update lower bound on SCP}
  end if
  if  $z(\mathbf{u}^k) > LB_{\text{best}}$  then
     $LB_{\text{best}} \leftarrow z(\mathbf{u}^k)$            {update best lower bound}
  else if  $z(\mathbf{u}^k) < LB_{\text{worst}}$  then
     $LB_{\text{worst}} \leftarrow z(\mathbf{u}^k)$            {update worst lower bound}
  end if
end while       {lower bound termination tolerance met}

```

#### 4. Greedy heuristic

In this section we present a simple heuristic used to project an infeasible vector into the feasible space of MCLS. Depending on the “goodness” of the Lagrangean vector, solutions built by the *Lagrangean Optimization* scheme can

be either feasible or infeasible. It is worth remembering that solutions to MCLS( $\mathbf{u}$ ) are built by separately solving  $P$  different ULS problems (one for each item) and joining together these solutions.

We define a greedy heuristic scheme that attempts to project the infeasible solution back to the feasible space of MCLS. Let us suppose that the capacity constraint of period  $t$  is not respected, which means the production plan suggested by the Lagrangean phase uses more capacity than available in period  $t$ , with  $t \in 1, \dots, T$ . The heuristic scheme attempts to restore feasibility by moving backward or forward a certain amount of production, until overload production of period  $t$  is eliminated.

Let us first consider the case in which we want to move backward production of a certain item  $j$  from period  $t$  to period  $t'$ , with  $t' < t$ . We need to compute the following quantities:

- amount of production of item  $j$  moved from period  $t$  to period  $t'$

$$\Delta_p(j)^{t'} = \min \left\{ b_{t'} - \sum_j (a_{jt'} y_{jt'} + m_{jt'} x_{jt'}), y_{jt}, 0 \right\};$$

- cost increase when production of item  $j$  is moved from period  $t$  to period  $t'$

$$\Delta_c^{t'}(j) = -\hat{f}_{jt} + f_{jt'}(1 - x_{jt'}) + (c_{jt'} - c_{jt})\Delta_p^{t'}(j),$$

where  $\hat{f}_{jt} = f_{jt}$  if  $\Delta_p^{t'}(j) = y_{jt}$  and  $\hat{f}_{jt} = 0$ , otherwise.

On the other hand, when attempting to move a certain amount of production from period  $t$  to period  $t''$ , with  $t'' > t$ , we should remember that at least the amount of production required to cover the demand of item  $j$  in each period  $k \in [t, t'' - 1]$  should still be produced in period  $t$ . For this reason, let us define:

- amount of production of item  $j$  moved from period  $t$  to period  $t''$

$$\Delta_p^{t''}(j) = \min \left\{ b_{t''} - \sum_j (a_{jt''} y_{jt''} + m_{jt''} x_{jt''}), y_{jt} - \sum_{k=t}^{t''-1} d_{jk}, 0 \right\};$$

- cost increase when production of item  $j$  is moved from period  $t$  to period  $t''$

$$\Delta_c^{t''}(j) = f_{jt''}(1 - x_{jt''}) + (c_{jt''} - c_{jt})\Delta_p^{t''}(j) - \sum_{k=t}^{t''-1} h_{jk}\Delta_p^{t''}(j).$$

The overall heuristic scheme is presented as pseudocode in Procedure *Greedy Heuristic*. It is worth remarking that such greedy scheme does not guarantee that a feasible solution be found. Whenever the heuristic scheme fails in finding such a solution, the Lagrangean solution is discarded and a new iteration of the subgradient method is started. For this reason, in order to reduce the amount of work required to project a Lagrangean vector back into the feasible space, we call the greedy scheme only after reaching a certain degree of convergence with the Lagrangean phase. More specifically, we begin applying the projection scheme only when the difference between the worst and best lower bound obtained by the Lagrangean phase in the last 20 iterations is less than 1%. This implies that the greedy scheme is used when “near optimal” Lagrangean multipliers have been found. Another reason why we want to use *Greedy Heuristic* sparingly is because of its cost in terms of computational time. It is easy to see that the computational complexity of such heuristic is  $\mathcal{O}(T^2P)$ .

**Algorithm 3.** Greedy Heuristic**Require:**  $\hat{x}, \hat{y}, \hat{s}$  not feasible**Ensure:**  $\hat{x}, \hat{y}, \hat{s}$  feasible, or NULL if feasible solution not found

```

while  $(\exists t \in [1, T] : b_t < \sum_j (a_{jt}y_{jt} + m_{jt}y_{jt}))$  do
  for all  $j \in \{1, \dots, P : x_{jt} = 1\}$  do
     $(j^*, t, t^*) \leftarrow \text{best } \Delta_c^{tk}(j)$  for all  $k \in [1, T]$     {find best swap}
  end for
  if best swap found then
    swap  $\Delta_p^{tt^*}(j^*)$  units of production of item  $j^*$  from period  $t$  to period  $t^*$ 
  else
    return NULL
  end if
  update capacity used in period  $t$     {reduce infeasibility}
end while

```

**5. Experimental plan and computational results**

In this section we present computational results of the algorithm on random generated instances of MCLS. The algorithm has been implemented in C++, compiled with the GNU g++ compiler with the -O2 option, and run on an Pentium 4 1.2 GHz Linux Workstation with 256 MB of RAM memory. Source codes and random instances mentioned in this section and used throughout the experiment can be downloaded at <http://www.ccm.itesm.mx/goal/>.

The experiment plan is aimed at collecting information about three different measures:

- solution quality: since we generate random instances, the optimal value of these instances is not known beforehand. For this reason, we also implemented a branch and cut code (B&C) that, whenever possible, solves the random generated instances to optimality. The branch and cut code has been implemented using the Cbc() module of the COIN-OR Library [32] and has been implemented in C++ on a Pentium 4 1.2 GHz Linux Workstation with 256 MB of RAM memory. We compiled the code using the GNU g++ compiler. The use of B&C provides insight about the quality of the heuristic solution compared with the global optimum.
- computational effort: we collect information about the time to best found solution, while keeping the algorithm running for a fixed amount of wall-clock time.
- robustness: we investigate what the effects of changes in the value of the parameters are. More specifically, we want to detect whether there is a correlation between two algorithmic parameters, namely  $N$  (CE population size) and  $\rho$  (CE quantile size), and the quality of the heuristic solution, for a given instance size. We perform a statistical analysis based upon the “response surface methodology” to draw conclusions about the existence of such correlation. The key question addressed in this section is: given the input size, in terms of number of items and number of periods, is it possible to determine what the value of parameters  $N$  and  $\rho$  should be in order to obtain the best feasible solution?

This section is organized as follows: first, we present the random scheme used to generate the instance test bed; next, we illustrate how the response-surface methodology has been used in order to find a good combination of factor levels for any given instance size. Finally, we conclude validating the statistical model by collecting results of the heuristic scheme using the values suggested by the response surface, and comparing them with the solutions provided by the branch and cut scheme.

**5.1. Response surface methodology**

In this section we present how we adapted the response surface methodology [33] to determine a good value of the CE parameters, namely  $N$ —population size—and  $\rho$ —quantile size, in terms of an algebraical polynomial model.

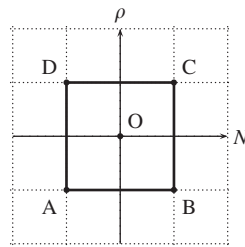


Fig. 1. Experimental design: experimentation phase.

Table 2

Results on benchmark problems from Trigeiro et al. [2] (average values)

No. items	No. periods	HFC	CQ	
		$\gamma$	$\gamma$	Time
6	15	2.87	0.93	1.31
6	30	1.95	0.65	1.44
12	15	0.84	0.07	2.09
12	30	0.74	0.71	2.05
24	15	0.31	0.42	1.93
24	30	0.28	0.11	1.87

Let us first illustrate the scheme used to randomly generate instances of MCLS. We provide as input the instance size (number of items and number of periods) and generate coefficients under the following scheme:

- $a_{jt} \in [5, 50]$ ,
- $c_{jt} = a_{jt} + 10$ ,
- $f_{jt} \in [3c_{jt}, 4c_{jt}]$ ,
- $h_{jt} \in [3, 20]$ ,
- $m_{jt} \in [2a_{jt}, 5a_{jt}]$ ,
- $d_{jt} \in [50, 100]$ ,
- $\gamma = \sum_j (d_{jt}a_{jt} + m_{jt})$ ,
- $b_t \in [0.5\gamma, 3\gamma]$ ,

where all the parameters have the meaning presented in Section 1.

We use the response-surface methodology with two factors, namely  $N$  and  $\rho$ , to find a good combination of factor levels for any given instance size. This phase of the experiment is divided into two parts:

- (1) experimentation phase: this phase can be seen as a preliminary study aimed at locating “promising” regions. First, an initial set of factor levels is chosen and, subsequently, using the *steepest ascent method*, the search is driven toward more attractive regions. The underlying assumption is that the surface can be approximated by a plane in the explored region;
- (2) exploration phase: once the first phase is concluded, we devote more attention to the region containing the optimal point. Using the *canonical analysis*, the surface is explored in more details, abandoning the planarity assumption and introducing higher order terms.

To begin with the first phase, we need to select an initial set of factor levels. Fig. 1 and Table 3 illustrate the initial configuration for the first phase of the experiment. Once we identify the region in which the initial search is to be performed, we run instances of different size with the five combinations of parameters illustrated in Table 3. More specifically, we run the algorithm on each instance for each combination of  $N$  and  $\rho$ . In Fig. 1,  $O = (500, 0.15)$ , and

Table 3  
Design center for phase I

	$N$	$\rho$
O	500	0.15
A	200	0.10
B	800	0.10
C	800	0.20
D	200	0.20

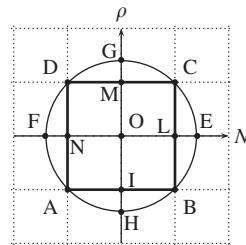


Fig. 2. Experimental design: exploration phase.

$A = (200, 0.1)$ , where for each pair the first number indicated the value of  $N$  and the second number provides the value of  $\rho$ . Next, we collect the objective function value provided by the algorithm on each run and test the following linear model:

$$Y_e = \beta_0 + \beta_1 N + \beta_2 \rho.$$

After computing the coefficients value of the regression model, we measured the significance level of the regression thought the ANOVA analysis. Using the steepest ascent method, we move the center of the experiment in the direction indicated by the gradient of  $Y_e$  and, eventually, we identify a region that contains a minimum of  $Y_e$ , which is, a region with a pronounced lack of fit to a plane. At this point, we move to the second phase of the methodology.

In the second phase, we test a fourth-order model using a robust composite design based upon the Box–Behnken design. We centered the experiment in the point  $N = 250, \rho = 0.15$ , with ranges of 300 for  $N$  and of 0.1 for  $\rho$ . However, due to the higher-order of the regression model, the number of combinations of factors tested must be increased. More specifically, the new design is presented in Fig. 2, where eight points—E, F, G, H, I, L, M, and N—have been added to the design.

Following the new design, we run the algorithm with the 13 different combinations of factor levels and, for each instance size, we collect the results in terms of objective function value. The process was repeated with different instance size, both in terms of number of items and number of periods. One interesting result observed from the ANOVA analysis of the fourth-order model is that the null hypothesis that states that the coefficients of  $\rho$  of any order are different from zero cannot be rejected. Consequently, it appears that  $\rho$  is not a meaningful parameter in the regression model, provided that its value is kept within the interval  $[0.1, 0.2]$ .

To illustrate, Fig. 3 shows how the objective function value of MCLS changes with respect to the value of  $N$ . In order to reduce the random error and to avoid bias related to the value of the coefficients of a particular instance, for every given problem size we randomly generate eight different instances. For this reason, each figure below reproduces eight different curves for a given problem size, each corresponding to a different instance. We run experiments with all the possible combinations of  $P$  and  $T$  values, where  $P$  is kept in the interval  $[5000, 15\,000]$  and varied with a step size of 2000, while  $T$  changes in the interval  $[12, 96]$  and modified with a step size of 12. For this reason, we define a total of  $6 \times 8 = 48$  different instance sizes. As mentioned before, for each of these 48 combinations, we randomly generated 8 different instances and, finally, on each of these instances we run the algorithm with 13 combinations of factor levels ( $N$  and  $\rho$ ). Consequently, we run the algorithm  $48 \times 8 \times 13 = 4992$  times, fixing a maximum computational time of 100 s per run. It is worth adding, thought, that such a design provides delimitations with respect to the validity of our

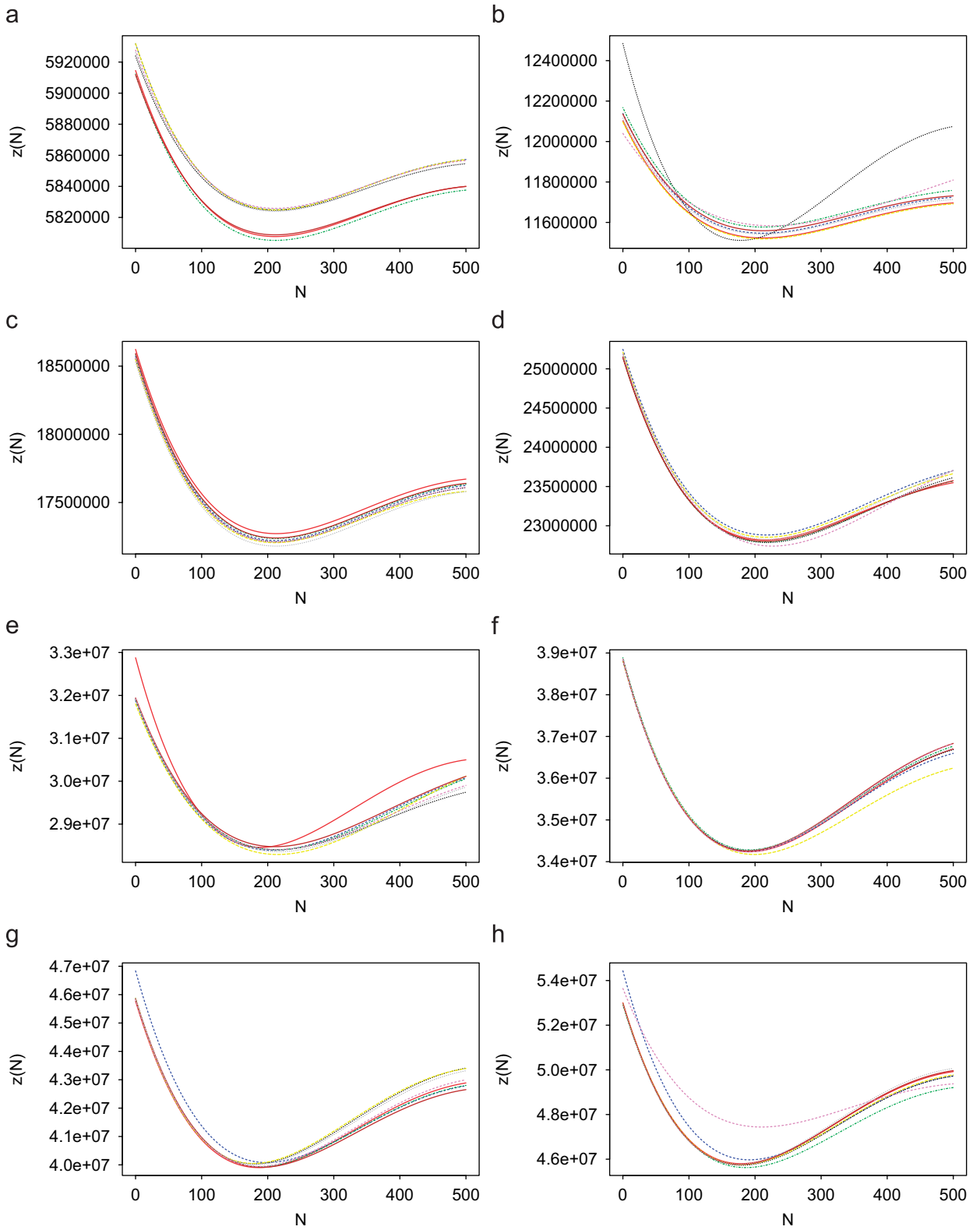


Fig. 3. Response surface with  $n = 1000$  items and (a)  $t = 12$  periods, (b)  $t = 24$  periods, (c)  $t = 36$  periods, (d)  $t = 48$  periods, (e)  $t = 60$  periods, (f)  $t = 72$  periods, (g)  $t = 84$  periods, (h)  $t = 96$  periods.

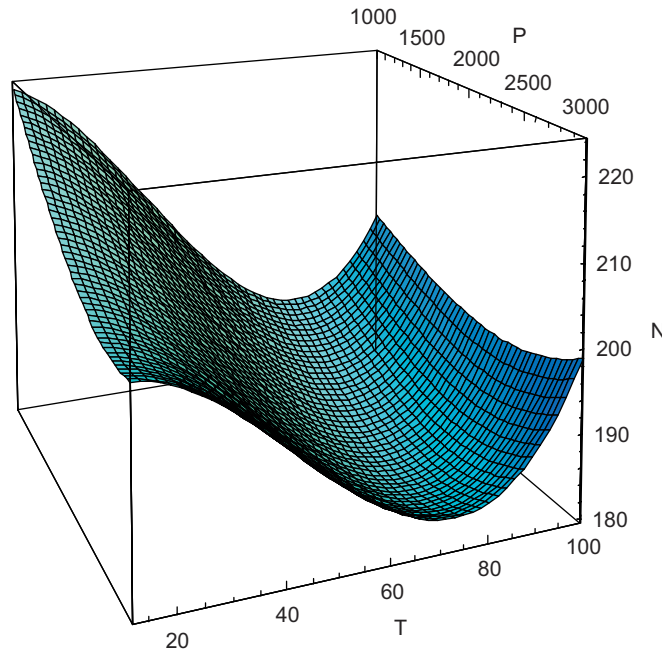


Fig. 4. Regression surface:  $N = f(P, T)$ .

Table 4  
Regression coefficients and  $p$ -values

	Estimate	Std. error	$p$ -value
$\beta_0$	2.521e + 02	8.767e + 00	< 2e – 16
$\beta_1$	–2.340e – 02	8.885e – 03	0.00915
$\beta_2$	–1.007e + 00	1.737e – 01	2.88e – 0.8
$\beta_3$	4.612e – 06	2.198e – 06	0.03727
$\beta_4$	6.535e – 03	1.570e – 04	4.82e – 05

results. The model we obtained is valid for problems with number of items and number of periods indicated above. However, such limitations still allow us to tackle very large scale problems.

Let us consider the case of a given instance size, say  $P = 1000$  and  $T = 12$  (Fig. 3(a)). After collecting the 13 observations of Fig. 2, we fit a fourth-order regression model such as

$$Y_e = \beta_0 + \beta_1 N + \beta_2 N^2 + \beta_3 N^3 + \beta_4 N^4$$

and we check, via ANOVA analysis, the significance level. Fig. 3(a) presents such curves for each one of the eight instances of size  $1000 \times 12$ . Next, in order to identify the value of  $N$  that gives the best objective function value for the given instance size, we minimize the polynomial  $Y_e$  by setting the first order derivative to zero and checking that the point is a local minimum. Fig. 3(b)–(h) presents the regression curve obtained for instances with 1000 items and a growing number of periods, from 24 to 96, respectively.

Using such a method, we collect  $48 \times 8$  minimum points, which are used to generate the final regression model that describes what the value of  $N$  should be for a given instance size in order to obtain the best solution with the proposed algorithm, e.g.,  $N = f(P, T)$ . The proposed regression model is

$$N = \beta_0 + \beta_1 n + \beta_2 t + \beta_3 n^2 + \beta_4 t^2.$$

Fig. 4 and Table 4 present the fitting surface as well as regression statistics for the model. As Table 4 shows, the coefficients of the polynomial are significant up to the second order and the model itself seems to be statistically significant, since the model has a  $p$ -value less than  $2.2e - 16$ .

An analysis of results highlights that there is a trade-off between intensification and diversification strategies: the surface  $N = f(P, T)$  of Fig. 4 illustrates that the optimal value of  $N$  decreases with the growth of  $P$  and  $T$ , even more so in the direction of the number of periods. This inverse correlation between  $N$  and  $T$  can be explained remembering that all tests were run for a fixed amount of computational time. When  $t$  grows, which is, when the size of the CE vector grows, increasing the population size corresponds to intensifying the search in a limited region of the feasible space, since the overall computational time is kept constant. On the other hand, reducing the population size allows to execute more CE iterations per unit time, this giving higher chances to the algorithm to diversify the search toward more attractive regions.

## 5.2. Computational results

In this section we finally present computational results that prove the effectiveness of the proposed algorithm. The value of the algorithmic parameters has been determined as illustrated in Section 5.1. This means that  $N$  is determined using the polynomial of Table 4 and  $\rho$  can take any value in  $[0.1, 0.2]$ . We fixed  $\rho = 0.1$  throughout the experiment.

We first illustrate the results of the algorithm on a subset of the 751 benchmark problems of Trigeiro et al. [2]. We also solved these instances using ILOG Cplex 10.0 Solver and we compared the solution obtained by our algorithm with the optimal solution of Cplex, if reached within the allotted time, or the best lower bound. On each instance we allowed Cplex to run for a maximum of 3000 s. In Table 2 we compare the results obtained by the proposed algorithm (column CQ) with those of [18] (column HFC). It is worth noting that the benchmark instances tested by these authors are small in size. Yet, they can still be used to compare our algorithm with other heuristic algorithms proposed in the literature. Computational time of HFC is only reported as average value over all the instances (5.84 s on a Pentium II 400 MHz). The gap indicator  $\gamma$  is computed as indicated below and the time is measured in wall clock time. The average gap of our algorithm is of 0.55%, compared with an average gap of 2.51% of HFC and of 3.2% of Gopalakrishnan et al. [20]. From this preliminary test, it is possible to observe that the proposed scheme further reduces the gap on these instances, hence performing better than previously proposed algorithms, while keeping the computational time below 3 s. Since different machines have been used, we cannot compare our computational time with that of Gopalakrishnan et al. [20] or Hindi et al. [18]. However, we can definitely conclude that our algorithm finds good quality solutions on a workstation in a short computational time.

To test the algorithm and validate the statistical model presented in the previous section, we generate a new set of instances, a testing set, different from the training set used during the calibration phase. Table 5 presents computational results on random generated instances of different size. The first and second columns define the instance, in terms of number of items and number of periods. The third column indicates the instance size in terms of number of variables, which is given by  $3PT$ . In the fourth column the chosen value of  $N$  is indicated, obtained using Table 4. Finally, the last two columns provide a measure of the effectiveness of the algorithm, in terms of computational time and solution quality. The quality of the solution is measured with respect to the solution obtained by running the branch and cut code for a fixed amount of computational time. More specifically,  $\gamma$  is computed as

$$\gamma = \begin{cases} \frac{z^H - z^*}{z^*} & \text{if global optimum is known,} \\ \frac{z^H - lb}{lb} & \text{otherwise,} \end{cases}$$

where  $z^H$  is the objective function value of the metaheuristic scheme,  $z^*$  and  $lb$  are the global optimum and the best lower bound obtained by B&C, respectively. In the  $\gamma$  score, we use the global optimum every time such a solution has been reached by B&C within the maximum computational time. On the other hand, if the branch and cut scheme does not terminate within the allotted time, we take as  $lb$  the best lower bound found by the scheme. We fix a maximum computational time of 300 wall-clock seconds for the metaheuristic scheme and of 10 000 wall-clock seconds for the branch and cut scheme. It is worth noting that  $\gamma = 0$  indicates that the metaheuristic scheme found the global optimum.

Table 5 proves the effectiveness of the proposed scheme. Small instances, in the range of 36 000–72 000 variables, for which the optimal solution has been reached by B&C, are always solved to optimality by the heuristic scheme in a short amount of computational time. On the other hand, on larger instances, for which no optimal solution has been found by B&C, the algorithm can only be measured with respect to the gap between the heuristic solution and the best

Table 5  
Results on random generated instances of MCLS

No. items	No. periods	No. vars	$N$	Time <sup>a</sup>	$\gamma^b$
500	24	36 000	222	15	0.0
	48	72 000	209	46	0.0
	72	108 000	203	126	0.52
	96	144 000	206	207	0.77
1000	24	72000	213	15	0.0
	48	144 000	201	87	0.78
	72	216 000	195	145	1.07
	96	288 000	197	167	1.21
2000	24	144 000	204	91	0.75
	48	288 000	191	156	1.20
	72	432 000	186	253	1.35
	96	576 000	188	298	1.48
5000	24	360 000	231	198	1.21
	48	720 000	218	271	1.76
	72	1 080 000	212	283	1.98
	96	1 440 000	215	279	2.31

<sup>a</sup>Wall-clock time.

<sup>b</sup>Expressed in percentage.

lower bound found by B&C. It can be noticed that the proposed scheme finds solutions whose gap is less than 2.5% of the best lower bound found by B&C. Furthermore, it is worth mentioning that, in all such large instances, the proposed scheme always finds a solution that is better than, or at least equal to, the best feasible solution found by B&C in 3000 s.

## 6. Conclusions

In this paper we present an hybrid algorithm for the capacitated lot sizing problem with setup cost and setup times. We develop a Lagrangean-based scheme that transforms the capacitated problem in a set of disjoint uncapacitated lot-sizing problems, which can be solved separately by applying an efficient CE-based algorithm. The CE algorithm is based upon a set of conditions expressed by a theorem, and is concerned with identifying which periods in the time horizon should be marked as production periods. Once the CE scheme indicates, for each item, when should this item be produced, it is easy to compute what the overall production and inventory cost associated to the policy would be. The CE-based primal scheme is embedded into a Lagrangean-based, dual scheme, aimed at finding near-optimal penalty costs for the capacity constraints.

We first calibrate the algorithm with a thorough statistical analysis, which lead to the identification of good values of algorithmic parameters. In order to find the value of such parameters, we used the response surface methodology with a total number of 4992 runs. Eventually, we applied a regression model that indicates, for a given instance size, in terms of number of items and number of periods, what the CE population size should be. The second phase of the experimental design was aimed at testing the algorithm on very large instances of the problem. We solved problems with more than one million variables in less than 300 wall-clock seconds with a maximum gap of less than 3%.

Considering future extensions to this work, the multi-facility version of the problem can be considered, where one needs to solve a capacitated lot-sizing problem over a number of facilities, each with a limited capacity. In addition, some tests regarding a parallel implementation of the algorithm will be carried out, to observe performances and scalability of the proposed algorithm.

## References

- [1] Aras OA. A heuristic solution procedure for a singlefacility multiitem dynamic lot sizing and sequencing problem. PhD thesis, Syracuse University; 1981.
- [2] Trigeiro WW, Thomas JL, McCain JO. Capacitated lot sizing with setups. *Manage Sci* 1989;35:141–61.

- [3] Nemhauser GL, Wolsey LA. Integer and combinatorial optimization. Wiley-interscience series in discrete mathematics and optimization. New York: Wiley; 1999.
- [4] Dzielinski M, Gomory R. Optimal programming of lot-sizing inventory and labor allocation. *Manage Sci* 1965;11:874–90.
- [5] Eppen G, Martin R. Solving multi-item capacitated lot sizing problems using variables redefinition. *Oper Res* 1987;35(6):832–48.
- [6] Pochet Y, Wolsey LA. Polyhedra for lot sizing with Wagner–Within costs. *Math Program* 1994;67:297–323.
- [7] Constantino M. A cutting plane approach to capacitated lot-sizing with start-up costs. *Math Program* 1996;75:353–76.
- [8] Miller AJ, Nemhauser GL, Savelsberg MW. Solving multi-item capacitated lot-sizing problems with setup times by branch and cut. *Core dp* 2000/39, Université Catholique de Louvain, Louvain-la-Neuve, Belgium; 2000.
- [9] Pochet Y. Valid inequalities and separation for capacitated economic lot sizing. *Oper Res Lett* 1988;7:109–16.
- [10] Miller AJ, Nemhauser GL, Savelsbergh MWP. On the capacitated lot sizing and continuous 0-1 knapsack polyhedra. *Eur J Oper Res* 2000;125:298–315.
- [11] Miller AJ, Nemhauser GL, Savelsbergh MWP. A multi-item production planning model with setup times: algorithms, reformulations, and polyhedral characterizations for a special case. *Core dp* 2001/6, Université Catholique de Louvain, Louvain-la-Neuve, Belgium; 2001.
- [12] Miller AJ, Nemhauser GL, Savelsbergh MWP. On the polyhedral structure of a multi-item production planning model with setup times. *Math Program Ser B* 2003;94:375–405.
- [13] Loparic M, Marchand H, Wolsey L. Dynamic knapsack sets and capacitated lot sizing. *Math Program* 2003;95(1):53–69.
- [14] Magnanti T, Sastry T. Facets and reformulations for solving production planning with changeover costs. *Oper Res* 2002;50(4):708–19.
- [15] Diaby M, Bahl HC, Karwan M, Zionts S. A Lagrangean relaxation approach for very large scale capacitated lot sizing. *Manage Sci* 1992;38:1329–40.
- [16] Tempelmeier H, Derstoff M. A Lagrangean-based heuristic for dynamic multilevel multi-item constrained lot sizing with setups. *Manage Sci* 1996;42:738–57.
- [17] Katok E, Lewis H, Harrison T. Lot-sizing in general assembly systems with setups costs, setup times, and multiple constrained resources. *Manage Sci* 1998;44:859–77.
- [18] Hindi KS, Fleszar K, Charalambous C. An effective heuristic for the CLSP with set-up times. *J Oper Res Soc* 2003;54:490–8.
- [19] Sambasivan M, Schmidt CP. A heuristic procedure for solving multi-plant, multi-item, multi-period capacitated lot sizing problems. *Asia-Pacific J Oper Res* 2002;19:87–105.
- [20] Gopalakrishnan M, Ding K, Bourjolly J, Mohan S. A Tabu-search heuristic for the capacitated lot-sizing problem with set-up carryover. *Manage Sci* 2001;47(6):851–63.
- [21] Caserta M, Quiñonez Rico E, Márquez Uribe A. A cross entropy algorithm for the knapsack problem with setups. *Comput Oper Res* 2007;35(1):241–52.
- [22] De Boer P, Kroese DP, Mannor S, Rubinstein RY. A tutorial on the cross-entropy method. *Ann Oper Res* 2005;134(1):19–67.
- [23] Rubinstein RY. Cross-entropy and rare events for maximal cut and partition problems. *ACM Trans Model Comput Simul* 2002;12(1):27–53.
- [24] Aggarwal A, Park JK. Improved algorithms for economic lot sizing problems. *Oper Res* 1993; 549–71.
- [25] Wolsey LA. Progress with single-item lot sizing. *Eur J Oper Res* 1995;86:395–401.
- [26] Karimi B, Fatemi Ghomi SMT, Wilson JM. The capacitated lot sizing problem: a review of models and algorithms. *Omega* 2003;31:365–78.
- [27] Brahimi N, Dauzere-Peres S, Najid NM, Nordli A. Single item lot sizing problems. *Eur J Oper Res* 2006;168:1–16.
- [28] Wagelmans A, Hoesel SV, Kolen A. Economic lot sizing: an  $\mathcal{O}(n \log n)$  that runs in linear time in the Wagner within case. *Oper Res* 1992;40(1):145–56.
- [29] Federgruen A, Tzur M. A simple forward algorithm to solve general dynamic lot sizing models with  $n$  periods in  $\mathcal{O}(n \log n)$  or  $\mathcal{O}(n)$  time. *Manage Sci* 1991;37:909–25.
- [30] Van Hoesel S, Kuik R, Salomon M, Van Vassenhove LN. The single item discrete lot sizing and scheduling problem: optimization by linear and dynamic programming. *Discrete Math* 1994;48:289–303.
- [31] Held M, Karp RM. The traveling salesman problem and minimum spanning tree: part II. *Math Program* 1971;1:6–25.
- [32] Lougee-Heimer R. The common optimization interface for operations research. *IBM J Res Develop* 2003;47(1):57–66.
- [33] Box G, Wilson K. On the experimental attainment of optimum conditions. *J R Stat Soc, Ser B* 1951;13:1–45.