The Multi-Item Capacitated Lot-Sizing Problem With Safety Stocks and Demand Shortage Costs

Nabil Absi[1*†], **Safia Kedad-Sidhoum**[2]

[1]Ecole des Mines de Saint-Etienne,

CMP - Site Georges Charpak - 880 Avenue de Mimet, F-13541 Gardanne, France

[2]Laboratoire d'Informatique de Paris 6,

4 place Jussieu, 75 252 Paris Cedex 05, France.

Abstract

We address a multi-item capacitated lot-sizing problem with setup times, safety stock and demand shortages. Demand cannot be backlogged, but can be totally or partially lost. Safety stock is an objective to reach rather than an industrial constraint to respect. The problem is NP-hard. We propose a Lagrangian relaxation of the resource capacity constraints. We develop a dynamic programming algorithm to solve the induced sub-problems. An upper bound is also proposed using a Lagrangian heuristic with several smoothing algorithms. Some experimental results showing the effectiveness of the approach are reported.

Keywords: Lot-sizing, dynamic programming, Lagrangian heuristic, safety stocks, shortages.

1 Introduction

The problem under consideration in this paper arises from industrial contexts. The production planning problems encountered in real-life situations are generally intractable due to a number of practical constraints. The decision maker has to find a good feasible solution in a reasonable execution time rather than an optimal one. The multi-item capacitated lot-sizing problem with setup times, demand shortage costs and safety stock deficit costs referred to as MCLSS, is a production planning problem in which there is a time-varying demand for a set of N items over T periods. The production should satisfy a restricted capacity and must take into account a set of additional constraints. Indeed, launching the production of an item i at a given period t for a demand requirement d_{it} involves a variable resource consumption v_{it} and a setup time f_{it}. The total available capacity at period t is c_t. For each period t, an inventory unitary cost γ_{it}^{+} is attached to each item i as well as a variable unitary production cost α_{it} and a setup cost β_{it}.

The problem has the distinctive feature of allowing demand shortages due to tight resource capacities. Indeed, when we are in lack of capacity to produce the total demand, we try to spread the available capacity among the items to minimize the total amount of demand shortages. Thus, we introduce in the model a unit cost parameter φ_{it} for item i at period t for the unmet amount of demand. These costs should be viewed as penalty costs and their values are very high in comparison with other unitary cost components.

*Corresponding author. E-mail: absi@emse.fr

†This work has been partially financed by DYNASYS S.A., under research contract no. 588/2002.

The use of safety stock is widely observed in industry to counter variability that may be present in a supply chain environment. In our study, a safety stock is an objective to reach rather than an industrial constraint to respect. When we cannot reach this target, we talk about safety stock deficits for which we introduce a unitary cost parameter γ_{it}^{-} for each item i at each period t. These costs should also be viewed as penalty costs and their values are lower than the shortage costs and very high in comparison with other cost components.

The MCLSS problem consists of finding a production plan that minimizes the demand shortages, the safety stock deficits as well as the setup, the inventory and the production costs allover the planning horizon. Chen and Thizy [8] proved that the multi-item capacitated lot-sizing problem with setup times is strongly NP-hard. There are many references dealing with the capacitated lot-sizing problem. We can quote the general review of Maes and Van Wassenhove [18] for heuristics developed to solve the problem, and the paper of Dogramaci *et al.* [10] that describes the so-called four-step heuristic. There are few references dealing with lot-sizing problems with shortage costs or safety stocks. Recently, Absi and Kedad-Sidhoum [2] proposed some Mixed Integer Programming (MIP) based heuristics to solve the MCLSS problem with additional constraints such as setup groups, operating sets, and lower and upper bounds on production. The same authors [3] developed also a branch-and-cut algorithm for the MCLSS problem without safety stocks. Aksen *et al.* [6] addressed the uncapacitated single-item lot-sizing problem with shortage costs. The authors proposed an $O(T^2)$ forward dynamic programming algorithm to solve the problem. Loparic *et al.* [17] proposed valid inequalities for the single-item uncapacitated lot-sizing problem with lower bounds on the stock variables where the objective is to maximize the sales. Diaby *et al.*[9] dealt with the lack of capacity by considering overtimes and Zangwill [25, 27] proposed a model with backlogs where the demands must be satisfied with an extra cost if delayed. We can also quote some related works dealing with the goodwill loss concept in lot-sizing literature. This concept was first mentioned by Hsu and Lowe [15]. The authors comment that production loss or customer goodwill loss may lead to a unit penalty cost for unsatisfied demand that grows in a nonlinear fashion, and is dependent on how long that demand has been back-ordered. Also inventory holding costs may be dependent on how long the items have been in stock. In [13], we come across goodwill loss again. The author formulates two models, one for the lost sales case, and another for the backorders. He assumes that the demand that cannot be met in a period is lost, and a loss of customer goodwill would manifest itself in terms of reduced future sales. A recent work of Aksen [5] improves the models, and proposes that a ratio of realized demand that goes unsatisfied in the current period be deducted from the original demand of the next period. Recently, Süral *et al.* [21] propose some efficient Lagrangian relaxation based heuristics for a lot-sizing problem with setup times where the objective is to minimize the total inventory carrying cost.

The main contribution of this paper is twofold. First, we develop a polynomial algorithm to solve the uncapacitated single-item lot-sizing problem with demand shortage and safety stock deficit costs (ULSS) optimally. Second, we develop a Lagrangian relaxation of the capacity constraints to obtain lower and upper bounds for the MCLSS problem.

An outline of the remainder of the paper is as follows. Section 2 describes a MIP formulation of the MCLSS problem. In Section 3, a Lagrangian relaxation of capacity constraints is presented. We state some properties of the optimal solutions of the ULSS problem, and we describe a polynomial algorithm to solve it. Section 4 illustrates a Lagrangian relaxation framework as well as a heuristic to construct feasible solutions for the MCLSS problem. Finally, computational results are given in Section 5 to show the effectiveness of the developed method based on a Lagrangian relaxation of capacity constraints.

2 Mathematical formulation and complexity

2.1 An aggregate formulation

In this section, we present a MIP formulation of the MCLSS problem. For a finite planning horizon of T periods and a number of items N, we set x_{it} as the quantity of item i produced at period t and y_{it} as a binary variable equal to 1 if item i is produced at period t (i.e. if $x_{it} > 0$) inducing fixed setup times and costs. The variable s_{it} for item i is the inventory value at the end of period t. The shortage for item i at period t is modeled by a nonnegative variable r_{it} with a high unit penalty cost φ_{it} in the objective function, the main goal is indeed to satisfy the customer and thus to have the minimum amount of demands not met.

Let s_{it}^+ and s_{it}^- represent respectively overstock and safety stock deficit variables of item i at period t. s_{it}^- has a high unit penalty cost γ_{it}^- in the objective function. However, this penalty is lower than the shortage cost. We set l_{it} as a parameter which represents the safety stock value of item i at period t. Thus, the stock of item i at period t is given by the expression $s_{it}^+ + l_{it} - s_{it}^-$. Moreover, we set $\delta_{it} = l_{it} - l_{i,t-1}$ which is the safety stock variation between two consecutive periods. It can be positive or negative. Furthermore, $\mathcal{I} = \{1, \cdots, N\}$ and $\mathcal{T} = \{1, \cdots, T\}$ denote respectively the set of items and the set of periods. The MCLSS problem is described by the following MIP:

$$\min \sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T}} \left(\alpha_{it} x_{it} + \beta_{it} y_{it} + \varphi_{it} r_{it} + \gamma_{it}^+ s_{it}^+ + \gamma_{it}^- s_{it}^- \right) \tag{1}$$

$$s.t. \tag{2}$$

$$s_{i,t-1}^+ - s_{i,t-1}^- + r_{it} + x_{it} = d_{it} + \delta_{it} + s_{it}^+ - s_{it}^-, \qquad \forall i \in \mathcal{I}, t \in \mathcal{T} \tag{3}$$

$$\sum_{i \in \mathcal{I}} (v_{it} x_{it} + f_{it} y_{it}) \leq c_t, \qquad \forall t \in \mathcal{T} \tag{4}$$

$$x_{it} \leq M_{it} y_{it}, \qquad \forall i \in \mathcal{I}, t \in \mathcal{T} \tag{5}$$

$$r_{it} \leq d_{it}, \qquad \forall i \in \mathcal{I}, t \in \mathcal{T} \tag{6}$$

$$s_{it}^- \leq l_{it}, \qquad \forall i \in \mathcal{I}, t \in \mathcal{T} \tag{7}$$

$$x_{it}, r_{it}, s_{it}^+, s_{it}^- \geq 0, \qquad \forall i \in \mathcal{I}, t \in \mathcal{T} \tag{8}$$

$$y_{it} \in \{0, 1\}, \qquad \forall i \in \mathcal{I}, t \in \mathcal{T} \tag{9}$$

The objective function (1) minimizes the total cost induced by the production plan that is production costs, inventory costs, shortage costs, safety stock deficit costs and setup costs. Constraints (3) are the inventory flow conservation equations through the planning horizon. Constraints (4) are the capacity constraints; the overall consumption must remain lower than or equal to the available capacity. If we produce an item i at period t, then constraints (5) impose that the quantity produced must not exceed a maximum production level M_{it}. M_{it} could be set to the minimum between the total demand requirement for item i on section $[t, T]$ of the horizon and the highest quantity of item i that could be produced regarding the capacity constraints, M_{it} is then equal to $\min \left\{ \sum_{t'=t}^{T} d_{it'}, (c_t - f_{it})/v_{it} \right\}$. Constraints (6) and (7) define upper bounds on respectively the demand shortage and the safety stock deficit for item i in period t. Constraints (8) and (9) characterize the variable's domains: x_{it}, r_{it}, s_{it}^+ and s_{it}^- are non-negative and y_{it} is a binary variable for $i \in \mathcal{I}$ and $t \in \mathcal{T}$.

2.2 A disaggregate formulation

Another formulation for the MCLSS problem is based on the facility location formulation initially proposed by Krarup and Bilde [16] for the uncapacitated single-item problem. A decision variable z_{irt} is used for the quantity of item i that is produced in period r to satisfy

a demand in period t. This implies that $\sum_{t=r}^{T} z_{irt} = x_{ir}$. The inventory of item i at period t is given by $\left(s_{it}^{+} + l_{it} - s_{it}^{-}\right)$ which is equal to $\sum_{t'=0}^{t} \sum_{t''=t+1}^{T} z_{it't''}$. We define a new cost parameter $\alpha'_{itt'} = \alpha_{it} + \gamma_{it}^{+} + \gamma_{i,t+1}^{+} + \ldots + \gamma_{i,t'-1}^{+}$ for $t \leq t'$. Using the variables z_{irt}, the MCLSS problem is :

$$\min \sum_{i \in \mathcal{I}, t \in \mathcal{T}} \left(\sum_{t'=t}^{T} \alpha'_{itt'} z_{itt'} + \beta_{it} y_{it} + \varphi_{it} r_{it} + (\gamma_{it}^{+} + \gamma_{it}^{-}) s_{it}^{-} - \gamma_{it}^{+} l_{it} \right) \tag{10}$$

$$s.t. \tag{11}$$

$$r_{it} + \sum_{r=1}^{t} z_{irt} = d_{it}, \qquad \forall i \in \mathcal{I}, t \in \mathcal{T} \tag{12}$$

$$\sum_{i \in \mathcal{I}} v_{it} \sum_{t'=t}^{T} z_{itt'} + \sum_{i \in \mathcal{I}} f_{it} y_{it} \leq c_t, \qquad \forall t \in \mathcal{T} \tag{13}$$

$$z_{irt} \leq M_{irt} y_{ir}, \qquad \forall i, t, r \leq t \tag{14}$$

$$r_{it} \leq d_{it}, \qquad \forall i \in \mathcal{I}, t \in \mathcal{T} \tag{15}$$

$$s_{it}^{-} \leq l_{it}, \qquad \forall i \in \mathcal{I}, t \in \mathcal{T} \tag{16}$$

$$s_{it}^{-} + \sum_{t'=0}^{t} \sum_{t''=t+1}^{T} z_{it't''} \geq l_{it}, \qquad \forall i \in \mathcal{I}, t \in \mathcal{T} \tag{17}$$

$$r_{it}, s_{it}^{-}, z_{irt} \geq 0, \qquad \forall i, t, r \leq t \tag{18}$$

$$y_{it} \in \{0, 1\}, \qquad \forall i \in \mathcal{I}, t \in \mathcal{T} \tag{19}$$

Constraints (12) ensure that the demand in period t is either satisfied by the production for period t performed in previous periods or the demands are not met. Constraints (13) are the capacity constraints; the overall consumption must remain lower than or equal to the available capacity. If we produce an item i in period t, then constraints (14) impose that the quantity produced must not exceed a maximum production level M_{irt}. M_{irt} could be set to the minimum between the demand requirement for item i at period t (d_{it}) and the highest quantity of item i that could be produced regarding the capacity constraints. M_{irt} is then equal to $\min\{d_{it}, (c_t - f_{ir})/v_{ir}\}$. Constraints (15) and (16) define upper bounds on respectively the demand shortage and the safety stock deficit for item i in period t. Constraints (17) ensure that the total quantity given by the inventory level and the safety stock deficit are greater than or equal to the safety stock level at period t. Constraints (18) and (19) characterize the variable's domains: r_{it}, s_{it}^{-} and z_{irt} are non-negative and y_{it} is a binary variable for $i \in \mathcal{I}$ and $t \in \mathcal{T}$. We also consider the following valid inequality that can strengthen the disaggregate formulation: $\sum_{r=t}^{T} z_{itr} \leq M'_{it} y_{it}$, where M'_{it} is equal to $\min\left\{\sum_{t'=t}^{T} d_{it'}, (c_t - f_{it})/v_{it}\right\}$ for an item i and a period t.

3 Lagrangian based lower bounds

3.1 Relaxing the resource capacity constraints

The general idea behind the Lagrangian relaxation approach is to decompose the multi-item capacitated lot-sizing problem into single-item uncapacitated lot-sizing sub-problems. This is possible by relaxing the resource capacity constraints (4) while using a set of Lagrange multipliers $\pi_t \geq 0$ in the objective function of the MCLSS model. We derive then a Lagrangian function denoted by $\mathcal{L}(x, y, r, s^{+}, s^{-}, \pi)$.

The Lagrangian relaxation of the capacity constraints of the MCLSS problem decomposes the model into N single-item uncapacitated lot-sizing problems with shortage and safety stock deficit costs which will be denoted as ULSS. Since we consider each item separately, it may be more convenient to discard the item index i to facilitate the reading of the following

mathematical model. The aggregate formulation of the ULSS problem is described in what follows:

$$\min \sum_{t\in\mathcal{T}} \left(\alpha_t x_t + \beta_t y_t + \varphi_t r_t + \gamma_t^+ s_t^+ + \gamma_t^- s_t^-\right) + \sum_{t\in\mathcal{T}} \pi_t \left(v_t x_t + f_t y_t - c_t\right) \tag{20}$$

$$s_{t-1}^+ - s_{t-1}^- + r_t + x_t = d_t + \delta_t + s_t^+ - s_t^-, \qquad \forall t \in \mathcal{T} \tag{21}$$

$$x_t \leq M_t y_t, \qquad \forall t \in \mathcal{T} \tag{22}$$

$$r_t \leq d_t, \qquad \forall t \in \mathcal{T} \tag{23}$$

$$s_t^- \leq l_t, \qquad \forall t \in \mathcal{T} \tag{24}$$

$$x_t, r_t, s_t^+, s_t^- \geq 0, \qquad \forall t \in \mathcal{T} \tag{25}$$

$$y_t \in \{0,1\}, \qquad \forall t \in \mathcal{T} \tag{26}$$

To the best of our knowledge, the ULSS problem was never addressed before in the literature. Therefore, we can recall that the model is an extension of the classical single item uncapacitated lot-sizing problem addressed by Wagner and Whithin [24]. Indeed, if we remove r_t and s_t^- from the model, it is solved in $O(T^2)$. This complexity was more recently improved [4], [11], [23]. Indeed, there are algorithms running in $O(TlogT)$ or even in linear time when costs are constant for all periods. We can notice that if we remove only the s_t^- variables, we obtain a model with shortage costs which is solved in $O(T^2)$ [6]. If we remove r_t, δ_t and the constraints (24), we obtain a model with backlogs which is also solved in $O(T^2)$ [27].

3.2 Fixed-charge network formulation

In this section, we propose a fixed-charge network model for the ULSS problem. The set of vertices is defined by the set of periods 1 to T of the problem. Vertex 0 is a dummy vertex that represents the principal source. Its supply is equal to $D = \sum_{t=1}^{T}(d_t + \delta_t) = \sum_{t=1}^{T} d_t + l_T - l_0$, with $l_0 = 0$.

X and R are two transshipment nodes which transfer respectively the total demand and the sum of the shortages. The remainder of the transshipment transits through node T, that is the maximum deficit on safety stock at the last period. It is obvious that the sum of the inputs at nodes X, R and T is equal to the total quantity which transits through node 0. For each node $t \in \{1, 2, \ldots, T\}$, we associate the quantity $\delta_t = l_t - l_{t-1}$. δ_t can be positive or negative. In the same way, nodes $\{1', 2', \ldots, T'\}$ represent the demands d_t in periods $t \in \{1, 2, \ldots, T\}$. The arcs of the network can thus be defined as follows:

- (X, t) are the production arcs for $t \in \{1, 2, \ldots, T\}$. The flow in arc (X, t) represents the quantity produced at period t,
- $(t, t+1)$ are the inventory arcs for $t \in \{1, 2, \ldots, T-1\}$ with a flow representing the ending stock of period t and the entering stock in period $t+1$,
- $(t, t-1)$ are the safety stock deficit arcs for $t \in \{2, \ldots, T\}$. The flow in these arcs represents the safety stock deficit of period $t-1$. The capacity of this arc is limited to l_t,
- the flow in arc (t, t') represents the demand that is met at period t for $t \in \{1, 2, \ldots, T\}$ and $t' \in \{1', 2', \ldots, T'\}$,
- the flow in arc (R, t') represents the shortage at period t for $t' \in \{1', 2', \ldots, T'\}$.

Figure 1 corresponds to a fixed-charge network representation of ULSS problem.

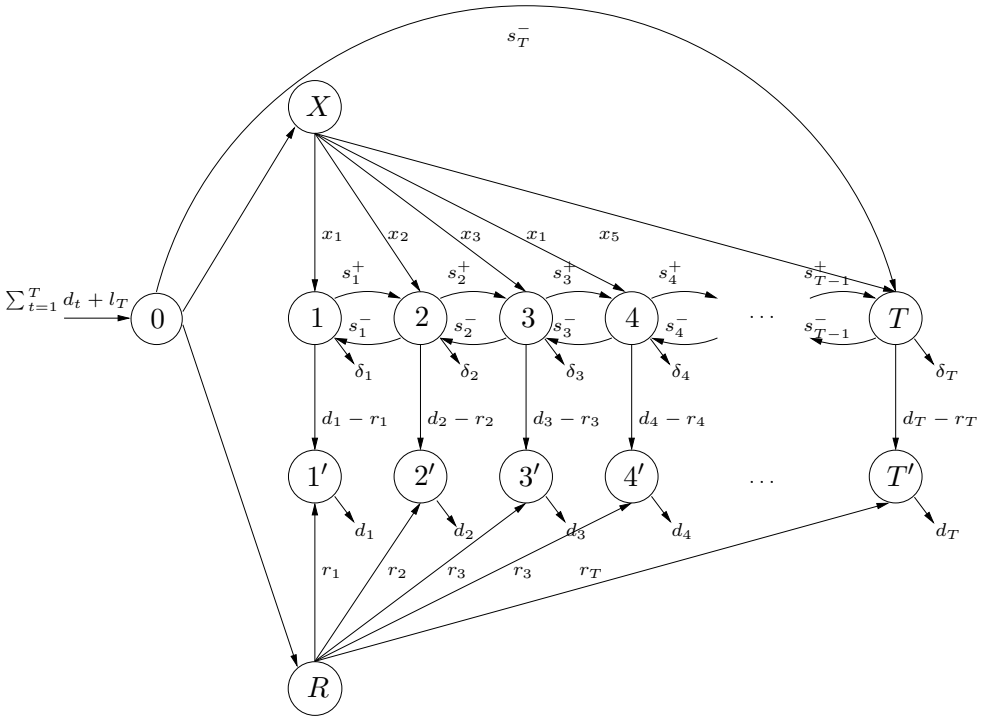

Figure 1: Fixed-charge network model for ULSS.

All the cost functions are concave. Indeed, the costs on arcs (t, t'), $(0, X)$ and $(0, R)$ are equal to zero. The arcs (R, t') have a unitary cost equal to φ_t. The arcs $(t, t+1)$ and $(t, t-1)$ have unitary costs equal respectively to γ_t^+ and γ_t^-. The cost on the production arcs (X, t) is equal to zero if the flow x_t is null and $\beta_t + \pi_t f_t + (\alpha_t + \pi_t v_t)x_t$ if the flow x_t is positive.

Finding an optimal solution to the ULSS problem is equivalent to finding a minimum cost extreme flow on the network depicted in figure 1. In the next section, we characterize the optimal solutions of the ULSS problem based on the fixed-charge network formulation.

3.3 Characterization of the ULSS optimal solutions

We present some properties of the ULSS optimal solutions. These properties enable us to propose a recursive dynamic programming algorithm for solving the problem in polynomial time in the worst case. We first begin with a definition.

Definition 1. *A feasible flow* $(\widehat{x}, \widehat{r}, \widehat{s^+}, \widehat{s^-})$ *is an extreme flow if there does not exist feasible flows* $(\widehat{x}_1, \widehat{r}_1, \widehat{s_1^+}, \widehat{s_1^-})$ *and* $(\widehat{x}_2, \widehat{r}_2, \widehat{s_2^+}, \widehat{s_2^-})$ *such that:* $(\widehat{x}, \widehat{r}, \widehat{s^+}, \widehat{s^-}) = (\widehat{x}_1, \widehat{r}_1, \widehat{s_1^+}, \widehat{s_1^-}) + (\widehat{x}_2, \widehat{r}_2, \widehat{s_2^+}, \widehat{s_2^-})$.

It is well known that if the objective function has a finite global minimum on the feasible region, then there must be an extreme flow which is an optimal flow [26]. Since the feasible region has a finite number of extreme flows, a search of all extreme flows will give the optimal flow. However, such a procedure might not be efficient. In what follows, we present some characteristics of the extreme flows of ULSS which will allow us to facilitate this search. The asterisk-superscripted variables in the lemmas and proofs signify optimal values.

The following lemma suggests that in an optimal solution, the overstock could not be strictly positive if we can not meet the level of the safety stock.

Lemma 1.
$s_t^{+*} s_t^{-*} = 0$ *for all* $t \in 1, \ldots, T$.

Proof. This result derives from the fact that an extreme flow uses a set of unsaturated arcs that represents a tree [24]. Hence, we cannot have $s_t^+ > 0$ and $s_t^- > 0$ since the costs on these arcs are non negative. The proof is made by contradiction. Let us suppose that we have $s_t^{+*} > 0$ and $s_t^{-*} > 0$ for a given period t at optimality. Then, decreasing s_t^{-*} to $(s_t^{-*} - \delta)$ and reducing s_t^{+*} to $(s_t^{+*} - \delta)$ maintains feasibility of the solution while lowering the total cost of the solution by $(\gamma_t^- + \gamma_t^+)\delta$. The solution could not be optimal which is a contradiction. □

In an optimal solution, we cannot have in a given period, demand shortage and production at the same time. More formally:

Lemma 2.
$r_t^* x_t^* = 0$ *for all* $t \in 1, \ldots, T$.

The proof uses the same arguments as Lemma 1 on the shortage and production variables. We next define several terms that will be useful to derive the dynamic programming algorithm.

Definition 2. *A period t is a* ***production point*** *if $x_t > 0$.*

Definition 3. *A period t is an* ***inventory point*** *if ($s_t^+ = 0$ and $s_t^- = 0$) or ($s_t^+ = 0$ and $s_t^- = l_t$). More precisely, a period t is an:*

- ***Inventory point of type 1*** *if $s_t^+ = 0$ and $s_t^- = 0$;*
- ***Inventory point of type 2*** *if $s_t^+ = 0$ and $s_t^- = l_t$.*

Using these production and inventory points, some properties will characterize the feasible extreme flows of the problem.

Proposition 1.
The following propositions are equivalent:

- $(\widehat{x}, \widehat{r}, \widehat{s^+}, \widehat{s^-})$ *is an extreme flow of ULSS;*
- $(\widehat{x}, \widehat{r}, \widehat{s^+}, \widehat{s^-})$ *is feasible and between every pair of production points (i, k) such that $i < k$, there exists at least an inventory point j such that $i \leq j < k$.*

Proof. Let us assume that we have two consecutive production points i and k with $i < k$ with no inventory point between them in an extreme flow. By Lemma 1, this implies that for all the vertices t between i and k we have either ($s_t^+ > 0$ and $s_t^- = 0$) or ($s_t^+ = 0$ and $0 < s_t^- < l_t$). This extreme flow contains a cycle where there is no edge flow fixed to its bound. If such a cycle exists in an extreme flow, then this flow can easily be expressed according to two other feasible flows. This proves that such a flow is not extreme and contradicts the assumption. Conversely, let us now assume that $(\widehat{x}, \widehat{r}, \widehat{s^+}, \widehat{s^-})$ is feasible, not extreme and between every pair of production points (i, k) such that $i < k$, there exists at least an inventory point j such that $i \leq j < k$. Since $(\widehat{x}, \widehat{r}, \widehat{s^+}, \widehat{s^-})$ is not extreme, the solution does not correspond to a maximal tree of the fixed-charge network. Therefore, the solution contains a cycle with no edge fixed to one of its bounds. Obviously, this cycle does

not contain inventory points, which is a contradiction. □

The following proposition is equivalent to the properties given by Proposition 1:

Proposition 2. $(\widehat{x}, \widehat{r}, \widehat{s}^+, \widehat{s}^-)$ *is feasible and between every pair of adjacent inventory points there exists at most one production point.*

The proof relies on the same arguments as the ones given in Proposition 1.

3.4 Numerical example of the ULSS problem

The following example illustrates an optimal solution structure for the ULSS problem. The data of the example are described in Table 1.

Period (t)	1	2	3	4	5
Demand (d_t)	1 000	1 000	1 000	1 000	1 000
Safety stock level (l_t)	200	800	1 000	1 000	800
Setup cost (β_t)	200 000	250 000	250 000	100 000	250 000
Unitary production cost (α_t)	70	70	70	70	70
Unitary inventory cost (γ_t^+)	55	55	55	55	55
Unitary shortage cost (φ_t)	250	212.5	175	137.5	100
Unitary deficit cost (γ_t^-)	150	127.5	105	82.5	60

Table 1: A five-period ULSS example.

The optimal solution of the problem is obtained using the mathematical model presented in Section 2.1 and the Cplex 9.0 Mixed Integer Linear Programming (MILP) solver. The solution is given in Table 2 and the associated network depicted in Figure 2.

Period (t)	1	2	3	4	5
Demand (d_t)	1 000	1 000	1 000	1 000	1 000
Safety stock (l_t)	200	800	1 000	1 000	800
Quantity produced (x_t)	2 800	0	0	2 000	0
Shortage (r_t)	0	0	200	0	0
Overstock (s_t^+)	1 600	0	0	0	0
Deficit on safety stock (s_t^-)	0	0	1 000	0	800

Table 2: Optimal solution of the ULSS example.

From Table 2, it is interesting to notice that the demand in period 3 is partially lost. We can also observe that period 2 and 4 (resp. 3 and 5) are inventory points of type 1 (resp. type 2). Figure 2 represents the network structure of the optimal solution. Arcs represented in bold are the ones fixed at their upper bounds which are the safety stock levels.

3.5 Dynamic programming algorithm

Based on the properties of the optimal solutions, we develop a dynamic programming algorithm to solve the ULSS problem. We have proved in the last section that a feasible solution is an extreme flow if and only if between every couple of adjacent inventory points i, k with $i < k$, there exists at most one production point j with $i < j \leq k$. A solution approach to find the optimal solution consists in searching all the extreme flows that satisfy this property and identify the minimum cost extreme flow. The rationale of the algorithm is to consider all the triplets (i, j, k) with $i < j \leq k$ where i and k are inventory points and j is a production point, and to find the sequence of triplets (i, j, k) which has the minimum

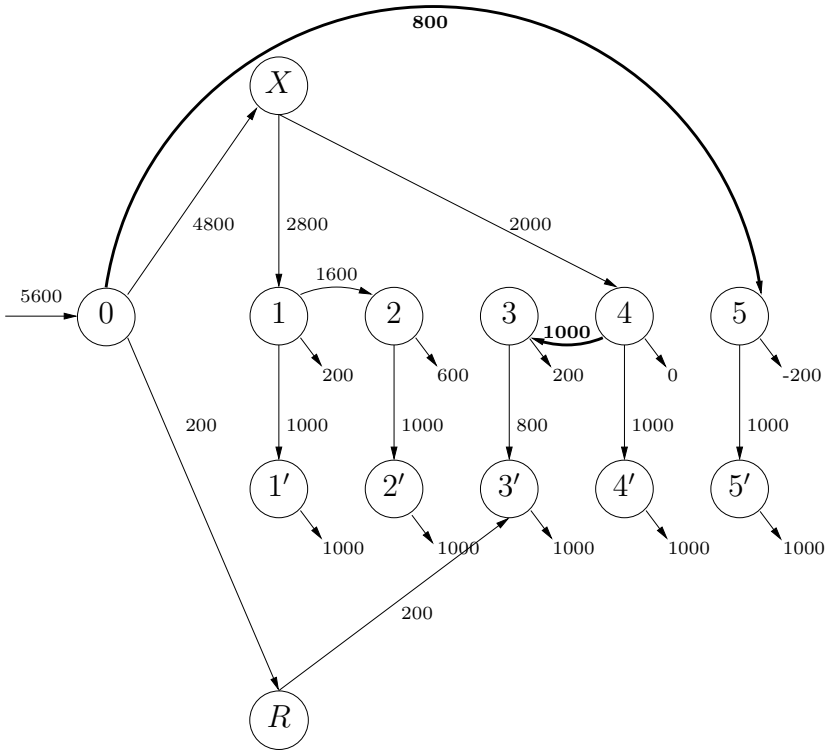

Figure 2: A network representation of the optimal solution of the ULSS example.

cost value. For each triplet (i, j, k) we define a sub-problem that we solve optimally. The sub-problem is a restricted problem of ULSS that will be denoted as ULSS_{ijk}^{uv}. The indices u and v characterize the inventory points (type 1 or 2). The index u (resp. v) is fixed to 0 if the period $i-1$ (resp. k) is an inventory point of type 1. The index u (resp. v) is fixed to 1 if the period $i-1$ (resp. k) is an inventory point of type 2.

If we assume that $\{i, \ldots, k\}$ represent the set of integers from i to k, ULSS_{ijk}^{uv} is defined on the interval of periods $\{i, \ldots, k\}$. There is a production point only at period j, the production variables at period $t \in \{i, \ldots, k\} \setminus j$ are equal to zero, i.e., $x_t = 0$. We denote with Z_{ijk}^{uv} the optimal value of the objective function of the problem ULSS_{ijk}^{uv}. ULSS_{ijk}^{uv} can be solved using a minimum cost flow algorithm.

The cost Z_{ik}^{uv} on the interval $i, \ldots, k$ can be computed by deriving the minimum between all the optimal policies on the interval $\{i, \ldots, k\}$ with j being the unique production period, $i < j \leq k$, and the optimal policy on the interval $\{i, \ldots, k\}$ with no production period which is denoted by Z_{i0k}^{uv}.

$$Z_{ik}^{uv} = \min \left[Z_{i0k}^{uv}, \min_{i<j\leq k} Z_{ijk}^{uv} \right] \tag{27}$$

An optimal basis for the ULSS solution corresponds to the minimum value of a sequence of adjacent inventory points intervals. The function F_k^u defines the minimum value of the adjacent inventory point sequence from period 1 to period k. This function can be computed using the following recursive formula:

$$F_k^v = \min_{i \leq k; u=0,1} \left\{ F_{i-1}^u + Z_{ik}^{uv} \right\} \tag{28}$$

The optimal value of ULSS problem is given by $F_T = \min_{v=0,1} F_T^v$.

The algorithm for solving the ULSS problem is described as follows:

Algorithm 1 solving ULSS

$u \leftarrow 0$, $v \leftarrow 0$
for all $t \in \{1, \ldots, T\}$ and $u \in \{0, 1\}$ **do**
  $F_t^u \leftarrow +\infty$, $F_0^u \leftarrow 0$
end for
for $i = 1$ to $T - 1$ **do**
  for $k = i + 1$ to T **do**
    repeat
      Compute Z_{i0k}^{uv}, $Z_{ik}^{uv} \leftarrow Z_{i0k}^{uv}$
      for $j = i + 1$ to k **do**
        Compute Z_{ijk}^{uv}, $Z_{ik}^{uv} \leftarrow \min \left[Z_{ik}^{uv}, Z_{ijk}^{uv} \right]$
      end for
      $F_k^v \leftarrow \min \{ F_k^v, F_{i-1}^u + Z_{ik}^{uv} \}$
      $v \leftarrow 1 - v$
      if $v = 0$ **then**
        $u \leftarrow 1 - u$
      end if
    until $u = 0$ and $v = 0$
  end for
end for
$F_T = \min_{u=0,1} F_T^u$

3.6 Algorithm complexity

The ULSS_{ijk}^{uv} problems can be solved using a minimum cost flow algorithm. One of the best strongly polynomial minimum cost flow algorithm is proposed by Orlin [20]. It solves the problem in $O(m \log n(m + n \log n))$, where m and n represent respectively the number of arcs and the number of nodes of the network.

In the worst case, the number of vertices of the problem network Z_{ijk}^{uv} is $2T + 3$ and the number of arcs is $5T+1$. Then, $n = 2T+3$ and $m = 5T+1$. The computation of Z_{ijk}^{uv} is done in $O(T^2 \log^2 T)$. The complexity of the algorithm that solves ULSS is thus $O(T^5 \log^2 T)$.

The complexity of $O(T^5 \log^2 T)$ is reduced to $O(T^2)$ when safety stocks are not considered. Aksen *et al.* [6] proposed an $O(T^2)$ dynamic programming algorithm to solve the problem optimally. This problem satisfies Zangwill conditions [26]. Indeed, at each period, either the demand at this period is totally lost, or entirely satisfied by a production or a stock. At a given period, we cannot observe a combination of the three possibilities, i.e., production, stock or shortage.

By relaxing constraints (24) of the ULSS problem and removing the safety stock variation parameters (σ_t), we obtain a single item lot-sizing problem with shortages and backlogging. When the capacity constraints on safety stock variables are relaxed, demands can also be met in future periods. This problem also satisfies Zangwill conditions [26]. Again, a demand in a given period can be satisfied by a production, a stock, a backlog or it will be totally lost. A combination of these four possibilities never occurs. According to this condition, it

is easy to derive an $O(T^3)$ algorithm to solve this problem optimally. The principle of this dynamic programming algorithm is based on the setting of the production points. Actually, an optimal solution can be found, just by looking for a sequence of production points that respect Zangwill conditions [25, 27] and minimize the total cost. This sequence takes $O(T^2)$ operations. The cost calculation for each consecutive production points takes $O(T)$. Thus, an $O(T^3)$ dynamic programming algorithm follows.

4 Lagrangian relaxation algorithm

The Lagrangian relaxation algorithm consists mainly of the following steps (we refer to Fisher [12] for a general survey on Lagrangian relaxation):

1. Initialization: all Lagrange multipliers π_t are set to 0.
2. For a given iteration k:
   (a) Solving the relaxed problem: compute $\mathcal{L}(x, y, r, s^+, s^-, \pi)$ at each iteration by using the dynamic programming algorithm 1 described in 3.5 for each ULSS_{ijk}^{uv} sub-problem. A current lower bound is found. If the lower bound value improves the current one, then it must be saved.
   (b) Lagrangian heuristic: a heuristic is used to find a feasible solution using a smoothing procedure as described in the following paragraph.
   (c) Updating Lagrange multipliers: use the subgradient optimization method to update the multipliers.
   (d) Stopping criteria: if any stopping condition is met, then save the best solution obtained.
   (e) Updating step length of the subgradient method: update the step length of the algorithm so that it satisfies the convergence conditions of the subgradient algorithm.

It is important to mention that the properties given by Lemma 1 and Lemma 2 are valid at each iteration of the algorithm. Indeed, between two consecutive iterations, the only variables affected by a change on their costs are the setup and the production variables.

In order to find a feasible solution at each step of the Lagrangian relaxation algorithm, we propose a Lagrangian based upper bound denoted as AKS. It is based on the Lagrangian solution obtained at each step of the Lagrangian relaxation process. Since the capacity constraints (4) are relaxed, the Lagrangian solution violates them. AKS heuristic is mainly based on a smoothing procedure to lower the deficits and shortages by reusing missing resource capacities. The heuristic is based on the work of Trigeiro *et al.* [22] who proposed an efficient Lagrangian relaxation heuristic for the classical multi-item capacitated lot-sizing problem with setup times. Recently, Brahimi *et al.* [7] proposed a generalization of Trigeiro *et al.* [22] smoothing heuristic to solve the capacitated lot-sizing problem with time windows.

AKS has a maximum of three passes and stops if the solution is feasible for the original problem or if it does not manage to remove overcapacity. In this case, a correction pass is used to remove unnecessary inventory. The smoothing heuristic AKS has the distinctive feature of obtaining a feasible solution at each step of the Lagrangian relaxation process. Basically, when AKS fails to find a feasible solution, it uses another pass to create either

safety stock or demand shortages. To select the item to be shifted at each step, different criteria were tested. Computational results show that the choice based on selecting the item in the increasing order of the ratio (29) given by the total Lagrangian cost divided by the quantity of removed overcapacity gives the best solutions. AKS smoothing heuristic is described in what follows:

1. *Backward pass:*

   Start at the last period and move production of a selected item to earlier periods to alleviate the capacity constraint violation. To select the item to be shifted, a ratio given by expression (29) is evaluated. Items are evaluated in the increasing order of this ratio until the overcapacity is eliminated.

$$(Q_{itt'}(v_{it'}\pi_{t'} - v_{it}\pi_t) + \pi_{t'}f_{it'} - \pi_t f_{it} + I_{itt'} + P_{itt'})/O_{itt'} \tag{29}$$

   where:

   - $Q_{itt'}$ represents the production of item i shifted from period t to period t'.
   - $I_{itt'}$ is the inventory cost derived from $Q_{itt'}$ when the production is shifted from period t to period t'. This cost takes into account inventory costs as well as safety stock deficit costs.
   - $P_{itt'}$ is the production cost induced by removing $Q_{itt'}$ at period t and producing $Q_{itt'}$ at period t'. This cost takes into account setup costs when needed.
   - $O_{itt'}$ is the amount of overcapacity removed at period t when shifting $Q_{itt'}$. Setup times are removed when needed.

2. *Forward pass:*

   Starting at the beginning of the planning horizon, production is shifted to the next later period. The criterion used is the minimization of the additional Lagrangian cost incurred per unit of shifted capacity. To select the item to be shifted, the expression (29) is used as previously except that t' is replaced by $t+1$. Items are evaluated in the increasing order of their ratio until capacity violation is eliminated.

3. *Shortage and fix-up pass:*

   When the setup variables are fixed according to previous passes, the model obtained is a linear programming model that can be solved polynomially. The model will create demand shortages and safety stock deficits when needed and will optimize the quantities produced and the inventory levels. The model will also remove unnecessary inventory. Obviously, the obtained solution is feasible for the original MCLSS problem.

5 Computational results

In this section, we present experimental results resulting from the application of the Lagrangian relaxation of the capacity constraints of the MCLSS problem. Our algorithms are implemented in the C++ programming language. We use the callable Cplex 9.0 library to solve the MILP problems. The computations were performed on an Intel Core 2 CPU 2.67 GHz PC with 3.25 GB RAM.

We have performed computational tests on a series of extended instances from the lot-sizing library LOTSIZELIB, initially described in [22]. These instances are denoted by

tr_{N-T}, where $N = 6, 12, 24$ is the number of items and $T = 15, 30$ is the number of periods. These instances are characterized by a variable resource consumption equal to one, and enough capacity to satisfy all demands over the planning horizon. They are also characterized, by important setup costs, small setup times and no safety stocks.

Since these instances have enough capacity to satisfy all demands over the planning horizon, we make some modifications to induce shortages. We have derived 72 new instances from the tr_{N-T} instances by increasing the resource requirements and adding some safety stocks. Variable resource requirements are multiplied by a coefficient $(1 + \eta)$ such that $0 \leq \eta \leq 0.001 \times c_t$ where c_t represents the available resource capacity at period t.

We carried out some modifications on setup times which are increased by multiplying them by a coefficient 1, 1.5, 2 or 3. Safety stock coverage is the number of periods where the amount of the demand is kept in the stock. Safety stock expressed in coverage is mainly used to cover the possible variation between forecasts and realized sales. At each period, safety stock is computed according to future needs. Thus, safety stock is equal to the sum of the demands on the section of the horizon which starts at this period and which has the length of the safety stock coverage. Safety stock coverage is fixed to 1, 1.5 or 2.

Demand and safety stock shortage costs are considered as penalty costs and their values must be higher than other cost components. Therefore, φ_{it} and γ_{it}^{-} are fixed such that $\varphi_{it} >> \gamma_{it}^{-} >> \max_{i',t'} \left\{\alpha_{i't'}; \gamma_{i't'}^{+}\right\}$. Moreover, shortage costs have the feature that they decrease over the horizon. Indeed, demands in the first periods of the horizon correspond to real orders and not forecasts in contrast to the demands in the last periods that are usually only predictions. They are generated in the same way for all the described instances.

The Lagrangian sub-problems obtained at each step of the Lagrangian relaxation are solved optimally using Cplex 9.0 solver for the model ULSS described in Section 3.1. However, it would be interesting to use a specific minimum cost flow algorithm. We use a subgradient method to compute the Lagrangian multipliers. The Lagrangian relaxation parameters are fixed according to the schema proposed by Held *et al.* [14].

We carried out a comparison between the following methods:

- the LR algorithm implements the Lagrangian relaxation of the capacity constraint which gives a lower bound for the MCLSS problem. At each step of the Lagrangian relaxation, an upper bound is computed using the Lagrangian heuristic described in the last section.

- the BC-AG method implements the standard branch-and-cut of Cplex 9.0 solver using the aggregate formulation of the MCLSS problem.

- the BC-FL algorithm implements the standard branch-and-cut of Cplex 9.0 solver using the facility location based formulation of the MCLSS problem.

In the LR and BC-AG algorithms, we used the aggregate model defined in Section 2.1 by the set of constraints (1)-(9). In the BC-FL algorithm, we used the disaggregate model defined by the set of constraints (10)-(19) in Section 2.2. Absi [1] shows that contrary to the classical multi-item capacitated lot-sizing problem, the aggregate formulation of the MCLSS problem leads to better linear relaxation bounds than its disaggregate formulation when using a standard solver.

The algorithm comparisons are based on the following criteria. The first one is the gap between the upper bound (UB) and the lower bound (LB). It is given by the formula suggested by Millar and Yang [19]:

$$\text{Gap} = 200 \times \frac{UB-LB}{UB+LB}$$

To evaluate the upper bound (UB) (resp. the lower bound (LB)), we define Gap_{UB} (resp. Gap_{LB}) as the distance from UB (resp. LB) to an estimated value of the optimal solution denoted by $\widetilde{opt}$. These gaps were previously used by Brahimi *et al.* [7]. For each instance, $\widetilde{opt} = (UB^* - LB^*)/2$ with UB^* and LB^* giving respectively the best upper bound and the best lower bound found so far for the problem. Thus, we have:

$$\text{Gap}_{UB} = 200 \times \frac{UB-\widetilde{opt}}{UB+opt} \text{ and } \text{Gap}_{LB} = 200 \times \frac{\widetilde{opt}-LB}{\widetilde{opt}+LB}$$

The LR algorithm stops if a maximum number of 100, 150, 200 and 300 iterations is reached or a time limit of 100, 300 or 600 seconds is achieved. We use the CPU time of the LR method as a stopping criterion (time limit) for the BC-AG and BC-FL methods. We also used a time limit of 100, 300 or 600 seconds as a stopping criterion for the BC-AG and BC-FL methods. To keep the presentation of the computational results concise, average gaps for all instances sharing the same value of a given parameter are reported on a single line. For example, the first gap of Table 3 represents the average gap of all instances having 6 items. Some detailed results will also be presented in Table 11.

Tables 3 and 4 summarize the computational behavior of the Lagrangian relaxation method. The stopping criterion is based on the maximum number of iterations. Average gaps are computed for each instance characterized by the following parameters: the number of items (N), the number of periods (T), the setup time factor and the safety stock coverage. Table 5 shows the average CPU time (s) of the Lagrangian relaxation method for the previous parameters.

		100 iterations			150 iterations		
		Gap	Gap_{LB}	Gap_{UB}	Gap	Gap_{LB}	Gap_{UB}
N	6	11.91%	5.87%	6.07%	7.53%	3.70%	3.83%
	12	6.92%	3.31%	3.62%	4.20%	2.01%	2.19%
	24	8.74%	4.42%	4.33%	5.53%	2.74%	2.80%
T	15	11.92%	5.91%	6.03%	7.33%	3.56%	3.78%
	30	6.46%	3.15%	3.32%	4.19%	2.08%	2.11%
Setup time factor	1	4.62%	2.20%	2.42%	4.25%	2.16%	2.09%
	1.5	12.95%	6.46%	6.51%	6.92%	3.32%	3.60%
	2	5.06%	2.46%	2.60%	4.77%	2.42%	2.34%
	3	14.19%	7.09%	7.13%	7.16%	3.39%	3.77%
Safety stock coverage	1	9.66%	4.83%	4.84%	6.47%	3.19%	3.28%
	1.5	8.70%	4.39%	4.32%	5.89%	2.91%	2.98%
	2	9.20%	4.37%	4.85%	4.91%	2.35%	2.57%

Table 3: Average gaps for the Lagrangian relaxation algorithm (100 and 150 iterations).

From Tables 3 and 4, we can notice that the difference between the gaps after 100 and 150 iterations is very big, while the difference between the gaps after 150 and 200 iterations is small, and the difference between 200 and 300 iterations is very small.

We can also notice that the gaps are higher when the number of items is smaller. The same remark is valid for the safety stock coverage and the number of periods. Indeed, when these parameters increase, all the gaps decrease. On the other hand, when the setup time factors is bigger, the gaps increase.

From Table 5, we can notice that the CPU time of the Lagrangian relaxation method is almost an increasing linear function of the number of iterations. One can observe that the execution time does not depend on the setup time factor and the safety stock coverage pa-

		200 iterations			300 iterations		
		Gap	Gap_{LB}	Gap_{UB}	Gap	Gap_{LB}	Gap_{UB}
N	6	7.15%	3.66%	3.49%	7.15%	3.66%	3.49%
	12	3.95%	1.99%	1.96%	3.93%	1.99%	1.94%
	24	5.30%	2.70%	2.60%	5.30%	2.70%	2.60%
T	15	6.88%	3.52%	3.37%	6.88%	3.52%	3.37%
	30	4.05%	2.05%	2.00%	4.03%	2.05%	1.99%
Setup time factor	1	4.25%	2.16%	2.09%	4.25%	2.16%	2.09%
	1.5	6.42%	3.28%	3.14%	6.42%	3.28%	3.14%
	2	4.76%	2.42%	2.34%	4.76%	2.42%	2.34%
	3	6.47%	3.29%	3.19%	6.44%	3.29%	3.16%
Safety stock coverage	1	6.22%	3.17%	3.05%	6.22%	3.17%	3.05%
	1.5	5.60%	2.85%	2.75%	5.58%	2.85%	2.73%
	2	4.58%	2.32%	2.25%	4.58%	2.32%	2.25%

Table 4: Average gaps of the Lagrangian relaxation algorithm (200 and 300 iterations).

		Number of iterations			
		100	150	200	300
N	6	50	129	225	471
	12	52	127	219	453
	24	54	138	240	495
T	15	50	134	236	493
	30	54	129	220	454
Setup time factor	1	33	66	108	227
	1.5	72	197	347	715
	2	36	67	109	229
	3	69	195	348	721
Safety stock coverage	1	29	72	124	260
	1.5	59	143	245	505
	2	69	179	315	654

Table 5: Computational results: Average CPU times (seconds) of the Lagrangian relaxation.

rameters. On the other hand, CPU times increase considerably when the number of periods increases, and they increase almost linearly when the number of items increases.

Computational results have shown that a number of iterations between 150 and 200 seems to be a good trade-off between the solution quality and the computation time. In what follows, we carried out computational experimentations in order to compare the results obtained by the Lagrangian relaxation method (LR) to those obtained by Cplex 9.0 standard solver (BC-AG) and (BC-FL). We focused on two scenarios where the number of iterations for the LR method are fixed to 150 and 200 iterations. The stopping criterion of the BC-AG and BC-FL methods is a time limit which is equal to the CPU time of the LR method. We also used a time limit criterion equals to 100, 300 and 600 seconds for the three methods.

Tables 6 summarizes the computational behavior of the BC-AG, BC-FL and LR methods based on 150 iterations stopping criterion of the LR method. Tables 7 summarizes the same computations for 200 iterations of the Lagrangian relaxation algorithm LR. Tables 8 summarizes the computational behavior of the BC-AG, BC-FL and LR methods based on a stopping criteria based on a CPU time equals to 100 seconds. Table 9 (resp. 10) summarizes the same computations with 300 seconds (resp. 600 seconds). Average gaps are calculated for different values of the following parameters: the number of items (N), the number of periods (T), the setup time factor and the safety stock coverage.

		LR method			BC-AG method			BC-FL method		
		Gap	Gap_{LB}	Gap_{UB}	Gap	Gap_{LB}	Gap_{UB}	Gap	Gap_{LB}	Gap_{UB}
N	6	8,94%	2,07%	6,87%	12,21%	10,19%	2,02%	30,06%	23,63%	6,56%
	12	5,46%	1,49%	3,97%	15,29%	13,85%	1,46%	26,52%	20,91%	5,70%
	24	2,87%	0,97%	1,90%	12,79%	11,73%	1,07%	24,50%	17,89%	6,70%
T	15	4,51%	0,97%	3,54%	7,09%	6,14%	0,95%	19,58%	14,34%	5,28%
	30	7,01%	2,05%	4,96%	19,77%	17,70%	2,08%	34,47%	27,28%	7,36%
Setup	1	3,50%	0,93%	2,57%	11,11%	10,11%	1,01%	27,76%	20,65%	7,23%
time	1,5	4,87%	1,29%	3,58%	13,00%	11,68%	1,33%	27,36%	20,99%	6,48%
factor	2	6,58%	1,74%	4,85%	14,44%	12,75%	1,70%	26,84%	21,05%	5,89%
	3	8,07%	2,08%	6,00%	15,17%	13,16%	2,03%	26,14%	20,56%	5,68%
Safety	1	7,53%	1,96%	5,58%	14,81%	12,89%	1,94%	26,07%	21,28%	4,87%
stock	1,5	5,53%	1,51%	4,02%	13,93%	12,42%	1,52%	28,11%	21,96%	6,26%
coverage	2	4,20%	1,06%	3,14%	11,54%	10,46%	1,09%	26,89%	19,19%	7,83%

Table 6: Computational results: the LR method vs. the BC-FL and BC-AG methods (150 iterations).

		LR method			BC-AG method			BC-FL method		
		Gap	Gap_{LB}	Gap_{UB}	Gap	Gap_{LB}	Gap_{UB}	Gap	Gap_{LB}	Gap_{UB}
N	6	8,61%	1,95%	6,66%	11,55%	9,66%	1,90%	29,23%	23,03%	6,33%
	12	5,08%	1,44%	3,65%	15,01%	13,60%	1,42%	25,92%	20,69%	5,31%
	24	2,70%	0,93%	1,77%	12,69%	11,61%	1,08%	22,14%	17,79%	4,39%
T	15	4,51%	0,92%	3,58%	6,77%	5,86%	0,91%	19,14%	13,94%	5,23%
	30	6,42%	1,95%	4,47%	19,39%	17,38%	2,03%	32,39%	27,06%	5,46%
Setup	1	3,36%	0,88%	2,47%	10,82%	9,84%	0,99%	26,30%	20,35%	6,05%
time	1,5	4,74%	1,21%	3,53%	12,63%	11,37%	1,27%	26,25%	20,71%	5,63%
factor	2	6,32%	1,69%	4,63%	14,11%	12,44%	1,69%	25,84%	20,75%	5,17%
	3	7,44%	1,97%	5,48%	14,76%	12,83%	1,93%	24,67%	20,20%	4,53%
Safety	1	7,15%	1,90%	5,25%	14,47%	12,57%	1,91%	25,48%	21,01%	4,54%
stock	1,5	5,30%	1,38%	3,91%	13,48%	12,06%	1,43%	27,15%	21,64%	5,61%
coverage	2	3,95%	1,03%	2,92%	11,29%	10,24%	1,06%	24,65%	18,86%	5,88%

Table 7: Computational results: the LR method vs. the BC-FL and BC-AG methods (200 iterations).

From Tables 6, 7, 8, 9, 10, we can notice that the LR method gives better lower bounds than the BC-AG and BC-FL methods. On the other hand, BC-AG gives better upper

		LR method			BC-AG method			BC-FL method		
		Gap	Gap_{LB}	Gap_{UB}	Gap	Gap_{LB}	Gap_{UB}	Gap	Gap_{LB}	Gap_{UB}
N	6	8,61%	1,86%	6,75%	10,99%	9,18%	1,82%	28,45%	22,22%	6,36%
	12	5,96%	1,56%	4,40%	15,27%	13,76%	1,52%	28,49%	20,74%	7,91%
	24	10,82%	4,03%	6,83%	12,95%	9,65%	3,32%	30,58%	15,78%	15,33%
T	15	4,51%	0,87%	3,64%	6,45%	5,60%	0,86%	18,70%	13,47%	5,27%
	30	12,42%	4,10%	8,35%	19,69%	16,13%	3,58%	39,64%	25,69%	14,47%
Setup	1	4,81%	1,24%	3,57%	10,73%	9,52%	1,21%	28,82%	19,89%	9,11%
time	1,5	6,97%	1,91%	5,07%	12,60%	10,77%	1,84%	28,25%	19,92%	8,48%
factor	2	9,46%	2,68%	6,78%	14,10%	11,55%	2,56%	33,19%	19,66%	14,17%
	3	12,60%	4,10%	8,55%	14,86%	11,62%	3,27%	26,44%	18,84%	7,71%
Safety	1	10,41%	2,93%	7,50%	14,45%	11,68%	2,78%	26,42%	19,98%	6,53%
stock	1,5	8,78%	2,84%	5,97%	13,50%	11,26%	2,25%	28,63%	20,68%	8,10%
coverage	2	6,19%	1,69%	4,51%	11,26%	9,65%	1,62%	32,47%	18,07%	14,97%

Table 8: Computational results: the LR method vs. the BC-FL and BC-AG methods (100 seconds).

		LR method			BC-AG method			BC-FL method		
		Gap	Gap_{LB}	Gap_{UB}	Gap	Gap_{LB}	Gap_{UB}	Gap	Gap_{LB}	Gap_{UB}
N	6	8,59%	1,76%	6,83%	9,90%	8,18%	1,72%	27,00%	20,85%	6,26%
	12	5,08%	1,41%	3,68%	14,71%	13,34%	1,39%	25,52%	20,34%	5,25%
	24	3,02%	1,01%	2,02%	12,69%	11,65%	1,05%	25,07%	17,78%	7,39%
T	15	4,51%	0,79%	3,72%	5,81%	5,03%	0,78%	17,72%	12,54%	5,21%
	30	6,62%	1,99%	4,63%	19,06%	17,09%	1,99%	34,00%	26,77%	7,40%
Setup	1	3,44%	0,85%	2,59%	10,23%	9,35%	0,88%	26,87%	19,56%	7,43%
time	1,5	4,81%	1,19%	3,62%	12,02%	10,82%	1,21%	26,49%	19,90%	6,70%
factor	2	6,37%	1,59%	4,79%	13,33%	11,78%	1,56%	25,71%	19,78%	6,02%
	3	7,63%	1,93%	5,70%	14,17%	12,28%	1,90%	24,37%	19,39%	5,06%
Safety	1	7,23%	1,84%	5,40%	13,69%	11,88%	1,82%	25,07%	20,15%	5,00%
stock	1,5	5,46%	1,35%	4,11%	12,83%	11,50%	1,34%	27,14%	20,84%	6,41%
coverage	2	4,00%	0,99%	3,01%	10,78%	9,80%	0,99%	25,37%	17,98%	7,50%

Table 9: Computational results: the LR method vs. the BC-FL and BC-AG methods (300 seconds).

		LR method			BC-AG method			BC-FL method		
		Gap	Gap_{LB}	Gap_{UB}	Gap	Gap_{LB}	Gap_{UB}	Gap	Gap_{LB}	Gap_{UB}
N	6	8,59%	1,70%	6,89%	9,25%	7,59%	1,67%	25,90%	19,89%	6,11%
	12	5,08%	1,38%	3,70%	14,39%	13,04%	1,36%	25,14%	20,05%	5,16%
	24	2,72%	0,90%	1,82%	12,53%	11,49%	1,05%	22,13%	17,63%	4,55%
T	15	4,51%	0,75%	3,76%	5,43%	4,69%	0,74%	17,07%	11,97%	5,12%
	30	6,42%	1,91%	4,52%	18,68%	16,72%	1,98%	31,70%	26,41%	5,42%
Setup	1	3,36%	0,78%	2,58%	9,90%	9,02%	0,88%	25,12%	19,10%	6,11%
time	1,5	4,71%	1,10%	3,61%	11,62%	10,45%	1,18%	25,09%	19,44%	5,73%
factor	2	6,35%	1,56%	4,79%	12,96%	11,44%	1,54%	24,16%	19,33%	4,89%
	3	7,44%	1,87%	5,57%	13,75%	11,92%	1,84%	23,19%	18,89%	4,36%
Safety	1	7,15%	1,74%	5,41%	13,20%	11,45%	1,76%	23,97%	19,65%	4,38%
stock	1,5	5,32%	1,29%	4,03%	12,46%	11,15%	1,32%	25,91%	20,39%	5,61%
coverage	2	3,93%	0,96%	2,97%	10,51%	9,52%	0,99%	23,29%	17,53%	5,83%

Table 10: Computational results: the LR method vs. the BC-FL and BC-AG methods (600 seconds).

bounds and BC-FL gives very bad lower bounds. Gaps provided by the LR method are better than those provided by the BC-AG and BC-FL methods. Some observations can be added as done for Tables 3 and 4. Indeed, when the number of periods increases, gaps increase for the BC-AG, BC-FL and LR methods. The same remark can be done for the setup time factor parameter except for the BC-FL method. For the BC-AG and BC-FL methods, we can notice that the difference between the gap of the 30 periods instances and those with 15 periods is very high. However, the BC-AG algorithm is less sensitive to the number of items. Basically, gaps of the LR method decrease when the number of items increases (except for Table 8), but gaps of the BC-AG and BC-FL methods do not depend on the number of items. We can also notice for the LR and BC-AG methods that the gaps are higher when safety stock coverage parameter decreases. This remark is not valid for the BC-FL method.

From Tables 6 to 10, we can easily notice that the BC-FL method gives the worst results especially for the lower bounds. Some detailed results presented in Table 11 confirm these observations. In fact, we can notice that lower bounds provided by the LR method are always better than those provided by the BC-AG and BC-FL methods. For instances with 6 items and 15 periods, the difference between the LR and BC-AG lower bounds is quite small. The same difference turns out to be very high for instances with 30 periods. On the other hand, the lower bounds attained by the LR method are always much tighter than those attained by the BC-FL method.

The LR method upper bounds are less interesting than those provided by the BC-AG method, but the difference is not very significant. For some instances with 24 items and 30 periods, the LR method outperforms the BC-AG algorithm. The BC-FL method gives the worst lower bounds. The upper bounds provided by BC-FL are generally less interesting, except for some instances when comparing the LR method to the BC-FL method.

According to Tables 6 to 11, we can say that the LR method gives better results than the BC-AG and BC-FL methods even if upper bounds are better for the BC-AG method. Therefore, we can easily notice that combining the BC-AG upper bounds and the LR lower bounds could lead to better gaps.

Parameters				LR method			BC-AG method			BC-FL method		
N	T	Setup time rate	Safety stock coverage	UB	LB	Gap	UB	LB	Gap	UB	LB	Gap
6	15	1.00	1.00	296565	279183	6,04%	281917	278614	1,18%	293780	244868	18,16%
6	30	1.00	1.00	334506	306699	8,67%	322876	274280	16,28%	336709	231883	36,87%
12	15	1.00	1.00	666281	643673	3,45%	648335	621604	4,21%	680548	552143	20,83%
12	30	1.00	1.00	738738	698517	5,60%	730081	579149	23,06%	750195	521152	36,03%
24	15	1.00	1.00	1179425	1164455	1,28%	1168001	1111768	4,93%	1223150	1008505	19,24%
24	30	1.00	1.00	1747192	1709233	2,20%	1756759	1471217	17,69%	1799446	1351046	28,47%
6	15	1.50	1.00	360977	326672	9,98%	333387	323285	3,08%	345168	288521	17,88%
6	30	1.50	1.00	416666	374550	10,65%	390917	332783	16,07%	417588	290841	35,78%
12	15	1.50	1.00	759602	733669	3,47%	741782	702087	5,50%	769137	635707	19,00%
12	30	1.50	1.00	909626	841276	7,81%	878574	692411	23,70%	905524	658441	31,60%
24	15	1.50	1.00	1358709	1326488	2,40%	1344316	1249054	7,35%	1388514	1141255	19,55%
24	30	1.50	1.00	2009067	1949292	3,02%	2023559	1650313	20,32%	2050633	1567970	26,68%
6	15	2.00	1.00	415047	365245	12,77%	383374	359105	6,54%	387732	330739	15,86%
6	30	2.00	1.00	500516	430676	15,00%	461651	383885	18,39%	471953	343038	31,64%
12	15	2.00	1.00	845476	808255	4,50%	820723	761779	7,45%	848185	701502	18,93%
12	30	2.00	1.00	1058708	961750	9,60%	1022146	796125	24,86%	1028889	775800	28,05%
24	15	2.00	1.00	1527898	1460191	4,53%	1485760	1371690	7,98%	1532714	1270592	18,70%
24	30	2.00	1.00	2245067	2149815	4,33%	2235905	1819293	20,55%	2257237	1755841	24,99%
6	15	2.00	1.50	507982	459858	9,94%	470396	449494	4,54%	492456	408221	18,70%
6	30	2.00	1.50	599693	532909	11,79%	560628	476263	16,27%	590585	404949	37,29%
12	15	2.00	1.50	1020203	985476	3,46%	998421	925965	7,53%	1039469	862329	18,63%
12	30	2.00	1.50	1251645	1167651	6,94%	1220557	963102	23,58%	1267830	902863	33,63%
24	15	2.00	1.50	1894410	1848300	2,46%	1867989	1730496	7,64%	1944961	1611699	18,74%
24	30	2.00	1.50	2669029	2582701	3,29%	2665954	2177051	20,19%	2705848	2052018	27,48%
6	15	3.00	1.50	574108	531623	7,68%	551685	520746	5,77%	566811	473017	18,04%
6	30	3.00	1.50	713550	634835	11,68%	669238	573017	15,49%	690658	485616	34,86%
12	15	3.00	1.50	1162901	1107950	4,84%	1134333	1036860	8,98%	1168741	973675	18,21%
12	30	3.00	1.50	1512339	1372617	9,69%	1440223	1121453	24,89%	1485615	1087662	30,93%
24	15	3.00	1.50	2172300	2080385	4,32%	2117783	1931931	9,18%	2192747	1821712	18,48%
24	30	3.00	1.50	3048650	2929187	4,00%	3034549	2454140	21,15%	3109165	2377886	26,65%
6	15	2.00	2.00	583906	552122	5,60%	564937	542274	4,09%	587462	509174	14,28%
6	30	2.00	2.00	691517	636338	8,31%	661520	579301	13,25%	695297	485644	35,51%
12	15	2.00	2.00	1193030	1159554	2,85%	1171468	1108326	5,54%	1239291	1051166	16,43%
12	30	2.00	2.00	1449080	1373652	5,34%	1421986	1142666	21,78%	1465826	1052612	32,82%
24	15	2.00	2.00	2261954	2229949	1,42%	2242053	2123503	5,43%	2355512	2032519	14,72%
24	30	2.00	2.00	3074359	3010255	2,11%	3077608	2576567	17,72%	3184653	2392486	28,41%
6	15	3.00	2.00	681565	622095	9,12%	640528	614508	4,15%	664068	572514	14,81%
6	30	3.00	2.00	793129	738336	7,16%	767869	673510	13,09%	796401	561463	34,60%
12	15	3.00	2.00	1354523	1287089	5,11%	1311066	1217646	7,39%	1356920	1147882	16,69%
12	30	3.00	2.00	1706693	1580876	7,65%	1651247	1307697	23,22%	1681761	1230676	30,98%
24	15	3.00	2.00	2548028	2476329	2,85%	2504506	2336407	6,94%	2615588	2204183	17,07%
24	30	3.00	2.00	3484747	3359647	3,66%	3471807	2855159	19,49%	3593773	2677518	29,22%
Mean gaps						6,11%			12,77%			24,65%

Table 11: Computational results (600 seconds).

6 Conclusion

In this paper, we propose a mathematical formulation of a new multi-item capacitated lot-sizing problem with setup times. This formulation takes into account several industrial constraints such as shortage costs and safety stock deficit costs. To derive a good lower bound, we have used an approach based on the Lagrangian relaxation of the resource capacity constraints. We have also proposed an algorithm that solves the induced single-item subproblems optimally by solving T^3 minimum cost flow problems where T denotes the number of periods. The complexity of this algorithm could be improved using a geometric approach [23]. It can also be improved by combining some new characteristics of the optimal solution. Some computational improvements could be reached by implementing a specific minimum cost flow algorithm. Numerical results show that our Lagrangian relaxation approach leads to better gaps and tighter lower bounds than those provided by a standard solver, while the standard solver gives better upper bounds. Our Lagrangian relaxation approach could be improved by developing a new heuristic or metaheuristic to construct better feasible solutions. Since the LR lower bounds are very good in comparison with the ones obtained by a standard solver, an exact solution method to be developed for the MCLSS in the future could leverage Lagrangian relaxation based lower bounds.

Acknowledgments

The authors would like to thank anonymous referees for their careful reading, thorough reviews and valuable comments that helped us improving the quality of the paper.

References

[1] N. Absi. Modélisation et résolution de problèmes de lot-sizing à capacité finie. *PhD thesis*, Université Pierre et Marie Curie (Paris VI), 2005.

[2] N. Absi and S. Kedad-Sidhoum. MIP-based heuristics for multi-item capacitated lot-sizing problem with setup times and shortage costs. *RAIRO - Operations Research*, 41(2):171–192, 2007.

[3] N. Absi and S. Kedad-Sidhoum. The multi-item capacitated lot-sizing problem with setup times and shortage costs. *European Journal of Operational Research*, 185 (3):1351–1374, 2008.

[4] A. Aggarwal and J.K. Park. Improved algorithms for economic lot size problems. *Operations Research*, 41:549–571, 1993.

[5] D. Aksen. Loss of customer goodwill in the uncapacitated lot-sizing problem. *Computers & Operations Research*, 34:2805–2823, 2007.

[6] D. Aksen, K. Altinkemer, and S. Chand. The single-item lot-sizing problem with immediate lost sales. *European Journal of Operational Research*, 147:558–566, 2003.

[7] N. Brahimi, S. Dauzère-Pérès, and N.M. Najid. Capacitated multi-item lot-sizing problems with time windows. *Operations Research*, 54 (5):951–967, 2006.

[8] W.H. Chen and J.M. Thizy. Analysis of relaxations for the multi-item capacitated lot-sizing problem. *Annals of Operations Research*, 26:29–72, 1990.

[9] M. Diaby, H.C. Bahl, M.H. Karwan, and S. Zionts. A Lagrangean relaxation approach for very large scale capacitated lot-sizing. *Management Science*, 38 (9):1329–1340, 1992.

[10] A. Dogramaci, J.C. Panayiotopoulos, N.R. Adam. The Dynamic Lot-Sizing Problem for Multiple Items Under Limited Capacity. *IIE Transactions*, 13 (4):294–303, 1981.

[11] A. Federgruen and M. Tzur. A simple forward algorithm to solve general dynamic lot sizing models with n periods in $o(n \log n)$ or o(n) time. *Management Science*, 37:909–925, 1991.

[12] M. L. Fisher. The Lagrangian relaxation method for solving integer programming problems. *Management Science*, 27:1–18, 1981.

[13] S.C. Graves. Manufacturing planning and control. In Oxford University Press, editor, *Handbook of applied optimization*, 2002.

[14] M. Held, P. Wolfe, and H.D. Crowder. Validation of subgradient optimization. *Mathematical Programming*, 6:62–88, 1974.

[15] V.N. Hsu and Lowe T.J. Dynamic economic lot size models with period-pair-dependent backorder and inventory costs. *Operations Research*, 49:316–321, 2001.

[16] J. Krarup and O. Bilde. Plant location, set covering and economic lot sizes: An O(mn)-algorithm for structured problems, in Optimierung bei graphentheoretischen and ganzzähligen Problemen. *L. Collatz et al. (eds), Birkhauser Verlag, Basel*, pages 155–180, 1977.

[17] M. Loparic, Y. Pochet, and L.A. Wolsey. The uncapacitated lot-sizing problem with sales and safety stocks. *Mathematical Programming*, 89:487–504, 2001.

[18] J. Maes and L. Van Wassenhove. Multi-Item Single-Level Capacitated Dynamic Lot-Sizing Heuristics: A General Review. *Journal of Operational Research Society*, 39(11):991–1004, 1988.

[19] H. H. Millar and M. Yang. Lagrangian heuristics for the capacitated multi-item lot-sizing problem with backordering. *International Journal of Production Economics*, 34:1–15, 1994.

[20] J.B. Orlin. A faster strongly polynomial minimum cost flow algorithm. *Operations Research*, 41(2):338–350, 1993.

[21] H. Süral, Denizel M., and L.N. Van Wassenhove. Lagrangean relaxation based heuristics for lot-sizing with setup times. *European Journal of Operational Research*, 2008.

[22] W. Trigeiro, L.J. Thomas, and J.O. McLain. Capacitated lot-sizing with setup times. *Management Science*, 35:353–366, 1989.

[23] A. Wagelmans, C.P.M. Van Hoesel, and A. Kolen. Economic lot sizing: an $o(n \log n)$ that runs in linear time in the Wagner-Whitin case. *Operations Research*, 40:S145–S156, 1992.

[24] H.M. Wagner and T.M. Whitin. A dynamic version of the economic lot size model. *Management Science*, 5:89–96, 1958.

[25] W.I. Zangwill. A deterministic multi-period production scheduling model with backlogging. *Management Science*, 13:105–119, 1966.

[26] W.I. Zangwill. Minimum concave cost flows in certain networks. *Management Science*, 14:429–450, 1968.

[27] W.I. Zangwill. A backlogging model and a multi-echelon model of a dynamic economic lot size production system - a network approach. *Management Science*, 15 (9):506–527, 1969.