8^{th} *International Conference of Modeling and Simulation - MOSIM'10 - May 10-12, 2010 - Hammamet - Tunisia*
"Evaluation and optimization of innovative production systems of goods and services"

HEURISTICS FOR THE MULTI-ITEM CAPACITATED LOT-SIZING PROBLEM WITH LOST SALES

Nabil Absi, Stéphane Dauzère-Pérès

École des Mines de Saint-Étienne
880 route de Mimet
F-13541 Gardanne France
absi@emse.fr, dauzere-peres@emse.fr

Boris Detienne

Université d'Avignon et des Pays de Vaucluse
339, chemin des Meinajaries
84911 Avignon Cedex 9 France
boris.detienne@univ-avignon.fr

ABSTRACT: *This paper deals with the multi-item capacitated lot-sizing problem with setup times and lost sales. Because lost sales, demands can be partially or totally lost. To find a good lower bound, we use a Lagrangian relaxation of the capacity constraints, when single-item uncapacitated lot-sizing problems with lost sales have to be solved. Each subproblem is solved using an adaptation of the $O(T^2)$ dynamic programming algorithm of (Aksen et al., 2003). To find feasible solutions, we propose a non-myopic heuristic based on a probing strategy and a refining procedure. We propose also some metaheuristic principles to improve solutions. Some computational experiments showing the effectiveness and limit of each approach are presented.*

KEYWORDS: *Lot-sizing, lost sales, Lagrangian relaxation, probing heuristics.*

1 INTRODUCTION

Production planning consists in deciding how to transform raw materials into final products in order to satisfy demands at time and minimal cost. Determining Lot-sizes is a crucial decision in production planning; which consists in calculating the quantity to produce for each item at each time period. In industrial contexts, several constraints may complicate the problem. In particular, the fact that items need a resource makes the problem more complex. Indeed, this can lead to the impossibility to entirely satisfy demands when there is not enough capacity. Such amount of unsatisfied requirement is referred to as *shortage on demand* or *lost sales*.

In this paper we address the single-level, single-resource, multi-item capacitated lot-sizing problem with setup times and lost sales called MCLS-LS. MCLS-LS consists in planning the production of N items over a horizon of T periods in order to satisfy time-varying demands. Demands are given for each item at each period. The different parameters of the problem are as follows. Producing a unit of item i in period t incurs a production cost α_{it} and requires v_{it} units of the total capacity C_t. The holding cost of a unit of product i at the end of period t is γ_{it}. Product dependent setup costs β_{it} and setup times f_{it} are incurred at each period where production takes place. The problem has the distinctive feature of allowing demand shortages (also called lost sales). Lost sales are particularly relevant in problems with tight capacities. Indeed, when the available resources are not sufficient to produce the total demand, we try to spread the capacity among the items by minimizing the total amount of lost sales. Thus, we introduce in the model a unitary lost sales cost φ_{it} for item i at period t. These costs should be viewed as penalty costs and their values are high in comparison with other unitary cost components. The objective is to minimize total production, setup, holding, and lost sale costs.

The classical multi-item capacitated lot-sizing problem with setup times (MCLS) is strongly NP-Hard (Chen and Thizy, 1990). Even the single item capacitated lot-sizing problem is NP-Hard in the ordinary sense (Bitran and Yanasse, 1982). The MCLS-LS problem is then NP-Hard. Lot-sizing problems have been studied for five decades, with numerous references dealing with capacitated lot-sizing problems. For a recent survey on lot-sizing problems, the reader can refer to (Jans and Degreaves, 2007, 2008) for multi-item capacitated lot-sizing problems, and (Brahimi et al., 2006) for single-item capacitated lot-sizing problems.

Different approaches were addressed in the literature in order to find near optimal solutions for the MCLS problem. (Trigeiro et al., 1989) were among the first to try to solve the MCLS problem. The authors propose a smoothing heuristic based on the Lagrangian relaxation of the capacity constraints. At each step of the subgradient method, a heuristic is called to obtain a feasible solution from the current Lagrangian

lower bound. (Belvaux and Wolsey, 2000), (Pochet and Wolsey, 1991) and (Miller et al., 2003) propose exact methods to solve MCLS problems by strengthening the Linear Programming (LP) relaxation using valid inequalities. Recently, (Degreaves and Jans, 2007) propose a new Dantzig-Wolfe reformulation and branch-and-price algorithm for the MCLS problem.

There has been little research on lot-sizing problems with demand shortages or lost sales. (Sandbothe and Thompson, 1990) introduced the concept of shortages to the classical uncapacitated lot-sizing problem, they called it stockouts. The authors propose an $O(T^3)$ dynamic programming algorithm. (Aksen et al., 2003) address the same problem but called this concept *lost sales*. They improved the previous complexity by proposing an $O(T^2)$ dynamic programming algorithm. (Berk et al., 2008) study the single-item lot-sizing problem for a warm/cold process with immediate lost sales by defining some properties of the optimal solutions. Recently, (Absi and Kedad-Sidhoum, 2008) propose a branch-and-cut algorithm to solve the MCLS-LS problem. They use a generalized version of (Miller et al., 2003) valid inequalities to strengthen the LP relaxations in the branch-and-bound tree. (Absi and Kedad-Sidhoum, 2007, 2009) develop respectively MIP-Based heuristics to solve the MCLS-LS problem with additional industrial constraints and a Lagrangian relaxation approach to solve the MCLS-LS problem with safety stocks.

The main contributions of this paper are twofold. First, we adapt the dynamic programming algorithm of (Aksen et al., 2003) in order to take any cost structure into account and use this algorithm in a Lagrangian relaxation approach to find good lower bounds. Second, we develop new Lagrangian heuristics based on a smoothing algorithm and a probing strategy to find near-optimal solutions. Generally, the smoothing heuristics that are presented in the literature are myopic. The algorithm proposed in this paper is non-myopic, a probing heuristic is used at each step to evaluate promising moves.

Section 2 describes MIP formulations of the MCLS-LS problem. In Section 3 we present a Lagrangian relaxation approach based on the relaxation of capacity constraints, an adaptation of the dynamic program proposed by (Aksen et al., 2003) to solve the single item uncapacitated version of MCLS-LS, and a subgradient method. In section 4 we present the principle of the Lagrangian heuristics based on a probing strategy as well as a refining procedure. In section 5 we present some computational experiments to show the effectiveness and limit of our procedures. Finally, Section 6 provides a short conclusion and future research directions.

2 MATHEMATICAL FORMULATION OF THE MCLS-LS PROBLEM

Different mathematical formulations were presented in the literature for the MCLS problem. In this section we recall one formulations of the MCLS-LS problem which is a generalization of the classical formulation of the MCLS problem usually called *aggregate formulation*. This formulation is addressed in several papers such as (Trigeiro et al., 1989). The notations are given below:

Sets and Indices
T: Number of periods. N: Number of items.
$\mathcal{I} = \{1, \cdots, N\}$. $\mathcal{T} = \{1, \cdots, T\}$.
i: index of an item, $i = 1, \ldots, N$.
t: index of a period, $t = 1, \ldots, T$.

Parameters:
d_{it}: demand for item i at period t.
C_t: available capacity at period t.
f_{it}: setup time for item i at period t.
v_{it}: unitary resource consumption for item i at period t.
α_{it}: production unit cost for item i at period t.
β_{it}: setup cost for item i at period t.
γ_{it}: inventory unit cost for item i at period t.
φ_{it}: lost sale unit cost for item i at period t.

Variables:
x_{it}: the amount of item i produced in period t.
y_{it}: binary setup variable, equal to 1 if item i is produced at period t (i.e. $x_{it} > 0$), and 0 otherwise.
s_{it}: the inventory value for item i at the end of period t.
r_{it}: the amout of lost sales of item i at period t.

The aggregate formulation of the MCLS-LS model is stated as follows:

$$\min \sum_{i \in \mathcal{I}, t \in \mathcal{T}} (\alpha_{it} x_{it} + \beta_{it} y_{it} + \varphi_{it} r_{it} + \gamma_{it} s_{it}) \quad (1)$$

$$s_{i,t-1} + r_{it} + x_{it} = d_{it} + s_{it}, \ \forall i, t \quad (2)$$

$$\sum_{i \in \mathcal{I}} (v_{it} x_{it} + f_{it} y_{it}) \leq c_t, \ \forall t \quad (3)$$

$$x_{it} \leq M_{it} y_{it}, \ \forall i, t \quad (4)$$

$$r_{it} \leq d_{it}, \ \forall i, t \quad (5)$$

$$x_{it}, r_{it}, s_{it} \geq 0, \ \forall i, t \quad (6)$$

$$y_{it} \in \{0, 1\}, \ \forall i, t \quad (7)$$

The objective function (1) minimizes the total cost that aggregates production, setup, inventory and shortage costs. Constraints (2) are the inventory balance equations. Constraints (3) are the capacity constraints; the overall consumption must be lower than or equal to the available capacity. Constraints (4) relate the binary setup variables to the continuous production variables. M_{it} can be set to

$\min\left\{\sum_{t'=t}^{T} d_{it'}, (c_t - f_{it})/v_{it}\right\}$. Constraints (5) define upper bounds on the lost sale variables. The domain definitions of the variables are defined in Constraints (6) and (7).

3 LAGRANGIAN BASED LOWER BOUNDS

When considering the MCLS-LS problem, capacity constraints (3) are the only constraints linking different items. By relaxing the capacity constraints and penalizing them in the objective function, we obtain a problem that is decomposed into N single-item uncapacitated lot-sizing problem with lost sales (ULS-LS). Under the assumption that the unitary lost sales costs are larger than the unitary production costs, ULS-LS can be solved using a T^2 dynamic programming algorithm (Aksen et al., 2003). In this section, we first present the resulting problem using the Lagrangian relaxation of capacity constraints, based on the agregate formulation. Second, we present an adaptation of the dynamic program proposed by (Aksen et al., 2003). Finally, we present a subgradient optimization algorithm to find a lower bound for MCLS-LS.

3.1 Lagrangian relaxation of capacity constraints

Since the seminal paper of (Trigeiro et al., 1989) on using Lagrangian relaxation for the classical MCLS problem, different authors used or improved this approach to address multi-item capacitated lot-sizing problems. The main idea behind the method is to decompose a multi-item capacitated lot-sizing problem into N single-item uncapacitated lot-sizing problems by relaxing the linking constraints (the capacity constraints in this case). The capacity violation is penalized in the objective function using Lagrange multipliers ($\pi_t \geq 0$).

When using Lagrangian relaxation on capacity constraints to derive a lower bound for the MCLS-LS problem, we obtain N ULS-LS problems that must be solved optimally. Since we consider each item separately, it is convenient to remove the item index i to facilitate the reading. The aggregate formulation of the ULS-LS problem is:

$$\min \sum_{t\in\mathcal{T}} (\alpha_t x_t + \beta_t y_t + \varphi_t r_t + \gamma_t s_t) + \sum_{t\in\mathcal{T}} \pi_t (v_t x_t + f_t y_t - c_t) \quad (8)$$

$$s_{t-1} + r_t + x_t = d_t + s_t, \ \forall t \quad (9)$$

$$x_t \leq M_t y_t, \ \forall t \quad (10)$$

$$r_t \leq d_t, \ \forall t \quad (11)$$

$$x_t, r_t, s_t \geq 0, \ \forall t \quad (12)$$

$$y_t \in \{0,1\}, \ \forall t \quad (13)$$

(Aksen et al. 2003) propose an $O(T^2)$ dynamic programming algorithm called ACC to solve the ULS-LS problem. The ACC algorithm works only if the assumption $\alpha_t > \varphi_t$ is true for all t. This assumption is used in (Aksen et al. 2003) to derive the following property.

Property 1 (Aksen et al., 2003)**.** *There is an optimal solution such that the demand at any period t will be fully satisfied if procurement is made during that period (i.e.* $x_t * r_t = 0$*).*

This property does not hold if the assumption $\alpha_t > \varphi_t$ is not satisfied. When performing a Lagrangian relaxation of the capacity constraints, the variables x_t are penalized in the objective function and this penalty increases the unitary production cost, which may become larger than the shortage cost, even if it is not the case in the original instance. The following counterexample shows that Property 1 is not valid in this case.

Consider the following ULS-LS problem with two periods. The parameters of the example are presented in Table 1.

Periods (t)	1	2
d_t	10	10
α_t	10	100
β_t	100	100
γ_t	0	0
φ_t	9	12

Table 1: Counterexample for the ULS-LS problem.

The details of the optimal solution for this counterexample is given in Table 2.

Periods (t)	1	2
x_t	10	0
y_t	1	0
r_t	10	0
s_t	10	0

Table 2: Optimal solution for the counterexample.

In this solution, we have $r_1 * x_1 = 100$, which contradicts Property 1. In the next subsection we propose ab adaptation of the recursive dynamic programming algorithm ACC.

3.2 Solving the ULS-LS problem

Let us present the ACC algorithm and its adaptation to take into account solutions that are no longer dominated when Property 1 is not valid. However, the other properties of dominant solutions are still valid ($s_{t-1} * x_t = 0, \ \forall t$, and $r_t * (d_t - r_t) = 0, \ \forall t$).

Definition 1. *Let us recall the definitions used in the ACC algorithm:*

- MC_{jt} *denotes the tradeoff between the marginal cost of losing the demand in period* t *and meeting this production with by a production in* j *before* t, $MC_{jt} = \min(\varphi_t, \alpha_j + \sum_{q=j}^{t-1} \gamma_q)$
- $C_j(t)$ *is the minimum cost of handling periods* $\langle 1, t\rangle$ *with* $s_t = 0$ *when period* j *is the last production period* ($j \leq t$).
- $C(t)$ *denotes the minimum possible cost of handling periods* $\langle 1, t\rangle$ *with* $s_t = 0$. $C(T)$ *is the optimal objective function, and it is computed by a forward recursion starting from* $C(0) = 0$.

The forward recursion of the ACC algorithm is recalled below:

Algorithm 1 The forward recursion of ACC

1: $C_0(t) \leftarrow \sum_{q=0}^{t} \varphi_q d_q, \forall t \in \langle 0, T\rangle$
2: **for** $t = 1$ to T **do**
3: $\quad C_t(t) \leftarrow C(t-1) + \beta_t + \alpha_t d_t$
4: $\quad$ **for** $j = 1$ to $t-1$ **do**
5: $\quad\quad C_j(t) \leftarrow C_j(t-1) + MC_{jt} d_t$
6: $\quad$ **end for**
7: $\quad C(t) = \min_{j \in \langle 0,t\rangle} \{C_j(t)\}$ and $j_t^* = \operatorname{argmin}_{j \in \langle 0,t\rangle} \{C_j(t)\}$
8: **end for**

In order to adapt the ACC algorithm to solve the general version of ULS-LS, we need to define the parameter $MC_{tt} = \min(\varphi_t, \alpha_t)$ and to replace Line 3 of the previous forward recursion by $C_t(t) \leftarrow C(t-1) + MC_{tt} d_t + \beta_t$.

To obtain the optimal policy of procurements and losses over the time horizon, we use an adaptation of the backtracking procedure of the ACC algorithm.

3.3 Lagrangian relaxation algorithm

The Lagrangian relaxation algorithm mainly consists of the following steps (see (Fisher, 1981) for a general survey on Lagrangian relaxation). The Lagrangian relaxation parameters are fixed according to the schema proposed by (Held et al., 1974).

1. Initialization: all Lagrange multipliers π_t are set to 0.
2. For a given iteration k:
   (a) Solve the relaxed problem using the dynamic programming algorithm described in 3.2 for each ULS-LS sub-problem. A current lower bound is found. If the lower bound value improves the current best one, then it is saved.
   (b) Update the Lagrange multipliers using the subgradient optimization method.
   (c) Stopping criteria: if any stopping condition is met, then save the best solution.
   (d) Update the step length of the subgradient method so that it satisfies the convergence conditions of the subgradient algorithm.

4 LAGRANGIAN BASED UPPER BOUNDS

Different methods were proposed in the literature to obtain good feasible solutions. The most interesting one was proposed by (Trigeiro et al., 1989) to solve the classical MCLS problem. To obtain a feasible solution, the authors developed a smoothing heuristic starting from a usually infeasible Lagrangian solution. In this section, we first present the general idea behind a general smoothing heuristic and the limits of this method. Then, we present the general scheme of a heuristic approach based on an improved smoothing procedure. The different components of our heuristic are finally detailed.

4.1 Smoothing heuristics principle

The general idea behind a smoothing heuristic is to start from the Lagrangian solution, which is most often infeasible, to construct a feasible solution. Production quantities are moved from one period to another in order to avoid capacity violations. (Trigeiro et al., 1989) propose a smoothing heuristic called TTM for the classical MCLS problem. The TTM smoothing heuristic is myopic, it performs slight modifications of the primal solution so as to fit the available capacity. TTM has a maximum of four passes and stops either when a feasible solution is obtained for the original MCLS problem or when the maximum allowed number of passes is reached (in this case, the procedure may not manage to remove over capacity). TTM starts with a backward pass that moves production quantities from the end to the beginning of the horizon to eliminate capacity violations. If the solution is still infeasible, a forward pass is used to move production from the beginning to the end of the horizon. The third and fourth passes are identical to the first and second passes. When a feasible solution is found, a fix-up pass is used to eliminated unnecessary inventory.

The advantage of this heuristic is its computational time which represents an average of less than one percent of the total computing time. Nevertheless, the TTM heuristic is myopic and does not consider the impact of moving a production quantity from one period to another. The decision of moving a production

is done a priori, based on the Lagrangian cost divided by the quantity of eliminated overcapacity. Since the heuristic computational time is very small, we propose in Section 4.3 a non-myopic heuristic to obtain a good upper bound for the MCLS-LS problem.

4.2 General scheme of our heuristic

In order to get good feasible solutions, we extend the approach of (Trigeiro et al., 1989) in a three-phase procedure:

Smoothing phase *(Improved_Smooth)* Starting from an optimal solution of the current Lagrangian subproblem (which usually does not respect the resource constraints), we try to reduce the overconsumption of the resource. See Section 4.3.

Cutting phase *(Cut)* This phase provides a feasible solution, starting from the potentially infeasible solution obtained at the smoothing phase. See Section 4.4.

Improving phase *(Improve)* This phase tries to improve the feasible solution by applying a hill-climbing approach. See Section 4.5.

The general organization of the heuristic is summarized in Algorithm 2.

Algorithm 2 Lagrangian heuristic

```
Let s_L be a Lagrangian solution
s* ← Cut(s_L)
s ← s_L ; iter ← 1
while s is not feasible and iter ≤ MaxIter do
  s ← Improved_Smooth(s)
  s̄ ← Cut(s)
  if cost(s̄) < cost(s*) then
    s* ← s̄
  end if
end while
if cost(s*) is smaller than the best known upper bound then
  s* ← Improve(s*)
end if
return s*
```

4.3 Non-myopic Smoothing heuristic

This part of our method is an improvement of the algorithm proposed by (Trigeiro et al., 1989). As already stated, in their paper, the authors propose to remove the overconsumption of the resource at a given period by moving production quantities to an earlier period (backward phase) or to a later period (forward phase).

Let us summarize the backward phase used by (Trigeiro et al., 1989) in Algorithm 3. *Greedy_Choice* returns an item i^* and a time period $t^* < t$ according to a greedy criterion which estimates the Lagrangian cost increase per unit of resource saved at period t. The forward phase is similar, except that the greedy choice is made among time periods after t.

Algorithm 3 Backward phase of Trigeiro et al. smoothing procedure $TrigeiroBackward()$

```
for t from T down to 1 do
  while resource constraint at t is violated do
    Greedy_Choice(t, i*, t*, backward)
    Move the correct amount of production quantity of i* from t to t*
  end while
end for
```

Although it is modulated by the Lagrangian multipliers, this criterion is very local since it considers only the current state of the plan and the state if the modification is made. The idea of our non-myopic heuristic is to use an adaptation of the algorithm of (Trigeiro et al., 1989) to estimate the quality of a move (*probing strategy*), instead of the initial criterion.

Algorithm 4 $Evaluate\_Move(t, i, t', direction)$

```
Move the correct production quantity of i from t to t'
if direction = Backward then
  Perform TrigeiroBackward
end if
Perform TrigeiroForward
Perform TrigeiroBackward
Perform the Cut heuristic
return the value of the solution obtained
```

This adapted version is summarized in Algorithm 4, and consists of a truncated run of the classical algorithm, followed by the cutting heuristic. It returns the cost of the solution that would be obtained if the classical criterion was applied until the end of the procedure after the considered move, thus an upper bound of the cost of the final solution after the move.

To design an improved backward or forward phase of the smoothing algorithm, we replace the use of *Greedy_Choice* by a procedure that finds the best item to move according to *Evaluate_Move*. Finally, an improved smoothing pass *Improved_Smooth()* consists in a improved forward pass followed by an improved backward pass.

One can notice that, although (Trigeiro et al., 1989) aim at obtaining a feasible solution at this step, this phase is, in our approach, devoted to finding solution

that is a good tradeoff between reducing infeasibility and minimizing costs.

4.4 Cutting phase

Unlike the classical MCLS problem, finding a feasible solution to MCLS-LS is easy: any solution that satisfies the resource constraints can be extended to a feasible solution by losing all unsatisfied demands. Moreover, if we assume that the production periods for each item are known (thus, the cost and resource consumption of the corresponding setups are fixed), then the resulting sub-problem consists in solving a linear program, that finds an optimal production quantities/lost sales plan consistent with the fixed setups.

Hence, the cutting phase of our algorithm is based on a simple variable-fixing method: for each item, the production periods in the starting solution (e.g. derived from the smoothing procedure) are fixed, while the others are forbidden. Formally, let $(\tilde{x}, \tilde{r}, \tilde{s})$ be the starting solution. The cutting heuristic sets the values $\bar{y}_{it} = 1$ if $\tilde{x} > 0$, and $\bar{y}_{it} = 0$ if $\tilde{x} = 0$, $\forall i, t$, and then solves the linear program (1-7) where variables y are fixed to the values of $\bar{y}$.

The resulting solution is then iteratively improved using the dominance Property 2 (in the sequel, this step is called *Refining procedure*). This property allows setups that were fixed but are not used at a profitable level in the current solution to be removed. It is used in a loop that solves the linear program according to the current set of fixed variables, applies the dominance property to try to remove some setups, and iterates until no setup variable has changed.

Property 2. *Let $(\bar{x}, \bar{y}, \bar{r}, \bar{s})$ be a feasible solution of (1-7) whose cost is $\bar{z}$, and let $i \in \mathcal{I}, t \in \mathcal{T}$. If $\bar{y}_{it} = 1$ and $\beta_{it} + \alpha_{it}\bar{x}_{it} > \varphi_{it}\bar{x}_{it}$, then there exists at least one feasible solution $(\bar{x}', \bar{y}', \bar{r}', \bar{s}')$ such that $\bar{y}'_{it} = 0$ and $\bar{y}'_{i't'} = \bar{y}_{i't'}$, $\forall (i', t') \in \mathcal{I} \times \mathcal{T} - \{(i, t)\}$, whose cost is smaller than $\bar{z}$.*

Proof. Straightforward: the solution where all variables are identical except $\bar{y}'_{it} = 0$, $\bar{x}'_{it} = 0$ and $\bar{r}'_{it} = \bar{x}_{it}$ is trivially feasible and has a cost smaller than $\bar{z}$. □

4.5 Improvement phase

This phase consists in a hill-climbing method with different neighborhoods. The idea of this approach is similar to the one used in the cutting phase: the set of setup variables is fixed, and the resulting LP is solved. As it is *easy* to compute an optimal solution corresponding to a vector of fixed variables, a solution is coded as its vector of setup variables. A neighbor of a solution is obtained by a slight change in this vector.

As some of the neighborhoods used in this method are relatively large, they are explored hierarchically, in order to reduce the computational effort needed for the search of an improving neighbor. More precisely, they are sorted according to an empirical evaluation of the ratio cost reduction expected on computational time required. If an improving solution is found in a given neighborhood, the remaining neighborhoods are not explored and the next iteration of the local search starts from the best solution found in the last neighborhood explored.

At each step, the neighborhoods are explored in the following order:

1. Try to invert one setup: for each $(i, t) \in \mathcal{I} \times \mathcal{T}$, define a neighbor by setting $\bar{y}'_{it} = 1 - \bar{y}_{it}$. The size of this neighborhood is $\mathcal{O}(N \cdot T)$.

2. Try to remove all the production of a given item at one period, and to balance it by creating a new setup for this item at another period: for each $i \in \mathcal{I}$, $t_1 \in \mathcal{T}, t_2 \in \mathcal{T} - \{t_1\}$ such that $\bar{y}_{it_1} = 1$ and $\bar{y}_{it_2} = 0$, define a neighbor by setting $\bar{y}'_{it_1} = 0$ and $\bar{y}'_{it_2} = 1$. The size of this neighborhood is $\mathcal{O}(N \cdot T^2)$.

3. Try to remove all the production of an item at a given period, and to use the released resource at this period for creating a new setup for another item: for each $i_1 \in \mathcal{I}$, $i_2 \in \mathcal{I} - \{i_1\}$, $t \in \mathcal{T}$ such that $\bar{y}_{i_1 t} = 1$ and $\bar{y}_{i_2 t} = 0$, define a neighbor by setting $\bar{y}'_{i_1 t} = 0$ and $\bar{y}'_{i_2 t} = 1$. The size of this neighborhood is $\mathcal{O}(N^2 \cdot T)$.

4. In a single move, try to move the production of a given item from a source period to a destination period, by removing its setup at the source period and creating a new setup at the destination period. Additional resource is released at the destination period by removing the existing setup of another item. Formally, for each $i_1 \in \mathcal{I}$, $i_2 \in \mathcal{I} - \{i_1\}$, $t_1 \in \mathcal{T}$, $t_2 \in \mathcal{T} - \{t_1\}$ such that $\bar{y}_{i_1 t_1} = 1$, $\bar{y}_{i_2 t_2} = 1$ and $\bar{y}_{i_2 t_1} = 0$, define a neighbor by setting $\bar{y}'_{i_1 t_1} = 0$, $\bar{y}'_{i_2 t_2} = 0$ and $\bar{y}'_{i_2 t_1} = 1$. The size of this neighborhood is $\mathcal{O}(N^2 \cdot T^2)$.

5. Try to create a new setup for a given item at a given period. Additional resource is released at this period by removing the existing setups of two other items at this period. Formally, for each $i_1 \in \mathcal{I}$, $i_2 \in \mathcal{I} - \{i_1\}$, $i_3 \in \mathcal{I} - \{i_1, i_2\}$, $t \in \mathcal{T}$ such that $i_1 < i_2$, $\bar{y}_{i_1 t} = 1$, $\bar{y}_{i_2 t} = 1$ and $\bar{y}_{i_3 t} = 0$, define a neighbor by setting $\bar{y}'_{i_1 t} = 0$, $\bar{y}'_{i_2 t} = 0$ and $\bar{y}'_{i_3 t} = 1$. The size of this neighborhood is $\mathcal{O}(N^3 \cdot T)$.

6. Try to swap the production periods of two items: for each $i_1 \in \mathcal{I}$, $i_2 \in \mathcal{I} - \{i_1\}$, $t_1 \in \mathcal{T}$, $t_2 \in \mathcal{T} - \{t_1\}$ such that $i_1 < i_2$, $\bar{y}_{i_1 t_1} = 1$, $\bar{y}_{i_2 t_2} = 1$, $\bar{y}_{i_1 t_2} = 0$ and $\bar{y}_{i_2 t_1} = 0$, define a neighbor by setting $\bar{y}'_{i_1 t_1} = 0$, $\bar{y}'_{i_2 t_2} = 0$, $\bar{y}'_{i_1 t_2} = 1$ and $\bar{y}'_{i_2 t_1} = 1$. The size of this neighborhood is $\mathcal{O}(N^2 \cdot T^2)$.

5 COMPUTATIONAL EXPERIMENTS

In this section, we present computational experiments resulting from the application of the Lagrangian relaxation of the capacity constraints of the MCLS-LS problem. Our algorithms are coded in C++ using Microsoft Visual Studio 2008. For the sake of comparison, we use the callable CPLEX 11.2 library to solve the MILP problems. In order to solve the linear programs in our heuristic methods, we use the Open Source Clp 1.11 solver from the COIN-OR library. The tests were done on an Intel Core Xeon CPU 2.5 GHz PC with 3 GB RAM under the Window XP operating system. CPU times are given in seconds.

We report computational tests on extended instances from the lot-sizing library LOTSIZELIB (Trigeiro et al. 1989). 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 unitary resource consumption equal to one, and enough capacity to satisfy all demands over the planning horizon. They are also characterized by important setup costs and small setup times. Based on these instances, we generated two new instances with 48 items and respectively 15 and 30 periods. Since these instances have enough capacity to satisfy all demands over the planning horizon, we made some modifications to induce shortages. We derived 32 new instances from the tr_{N-T} instances by increasing the resource requirements. Unitary resource consumptions are multiplied by $(1 + \eta)$ 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 1, 1.5, 2 or 4.

Lost sales costs are considered as penalty costs and their values must be larger than other cost components. Therefore, φ_{it} is fixed such that $\varphi_{it} >> \max_{i',t'} \{\alpha_{i't'}; \gamma_{i't'}\}$. We carried out a comparison between the following methods:

- the CPLEX method implements the standard branch-and-cut of CPLEX 11.2 solver using the aggregate formulation of the MCLS-LS problem.
- the LR algorithm implements the Lagrangian relaxation of capacity constraints which gives a lower bound for the MCLS-LS problem. During the Lagrangian relaxation, an upper bound is computed using the non-myopic smoothing heuristic described in the last section. This smoothing heuristic has three parameters. The first parameter (*Prob*) corresponds to the number of improved smoothing passes performed by the method (*probing depth*). The remaining passes (up to a total of four passes) are classical passes from the TTM algorithm. The second parameter (*Refine*) corresponds to activating or deactivating the refining procedure during the Lagrangian relaxation. The last parameter (LS) corresponds to activating or deactivating the local search described in the previous section.

To keep the presentation of the computational results concise, average gaps for all instances sharing the same value of given parameters are reported on a single line.

5.1 Impact of probing depth using the probing procedure

In this subsection, we present the impact of the depth parameter when using the probing procedure. Table 3 summarizes the computational behavior of the LR method by varying the depth of the probing heuristic (0, 1, 2 or 3). Average gaps (gap) and average CPU times (time) are computed for each set of instances characterized by the following parameters: A number of items (N) and a number of periods (T). The LR method stops if a time limit of 10000 seconds is achieved or the stopping criteria of the subgradient algorithm are satisfied. In these computations, the refining procedure is activated and the local search is not activated. In this subsection, gaps are calculated as the percentage difference between the best upper bound and the best lower bound compared to the best upper bound.

From Table 3, we note that the differences between the gaps of LR with probing depths 1 and 2 and LR without probing are rather large, while the differences between the gaps of LR with probing depth of 2 and probing depth of 3 are relatively small. We can observe that the computational time of the LR method with probing depth of 3 is relatively large compared to the other configurations. Note also that LR with a probing depth of 2 gives the best trade-off between the gap and the CPU time.

5.2 Impact of refining and local search procedures

In this subsection, we discuss the advantage of using the refining and local search procedures. Recall that the refining procedure uses the dominance property (Property 2) to improve the obtained solution.

Tables 4 summarizes the computational behavior of the LR method without improvement procedures, when only the refining procedure is activated and when both the refining and local search procedures are activated. In these computations, the depth of the LR method is fixed to 2. The stopping criterion is the same as in the previous subsection. Average gaps and average CPU times are computed for each set of instances characterized by the following parameters: the number of items (N) and the number of periods (T).

From table 4, it can be noted that, when activating the refine procedure, gaps are considerably improved whereas CPU times do not increase very much. We can also observe gap reductions when activating the local search procedure, but CPU times increase considerably for instances with 30 periods.

5.3 Quality of Lagrangian relaxation lower bounds

Table 5 shows a comparison between the Lagrangian relaxation lower bounds (LB_{LR}) and the best lower bound (LB_{CPLEX}) obtained after 1 000 seconds of the CPLEX branch-and-cut algorithm, based on the average ratio between (LB_{LR}) and (LB_{CPLEX}), and the average CPU time of the LR method. The LR algorithm stops if a time limit of 1 000 seconds is achieved or the stopping criteria of the subgradient algorithm are satisfied. In these computations, the improvement procedures are not activated.

N	T	LB_{LR}/LB_{CPLEX}	time (LR)
6	15	0,938	0,1
	30	0,971	0,3
12	15	0,998	0,2
	30	1,099	0,6
24	15	1,008	0,3
	30	1,069	1,1
48	15	1,009	0,7
	30	1,052	2,0
Mean		**1,018**	**0,7**

Table 5: Quality of LR lower bounds.

Table 5 shows that the LR method provides better lower bounds with an average CPU time that is less than one second, compared to the 1 000 seconds execution time of CPLEX. CPLEX only obtains better lower bounds for small instances (with 6 items).

5.4 Repeated subgradient procedure

Since the LR Method obtains good lower bounds within relatively small CPU times, we developed a new variant of the LR approach that we call multi-start Lagrangian relaxation approach (denoted LR_MS). Contrary to the LR method that stops when the subgradient conditions are satisfied, the LR_MS method restarts with new Lagrangian multipliers. These new multipliers are calculated from the last obtained multipliers increased by randomly generated values. This method stops when a given CPU time is reached.

Tables 6 and 7 summarize the computational behavior of LR_MS method with and without the local search procedure (LS) for different probing depths (0, 1, 2, or 3) and CPLEX methods subject to a 1000-second time limit. Tables 6 (resp. Table 7) reports average gaps that are computed for each set of instances characterized by a number of items (N) and a number of periods (T) (resp. by the same setup time factor (1, 1.5, 2 or 4)).

From Tables 6 and 7, it can be noted that the LR_MS method with local search and probing depths of 1 and 2 obtains the best gaps. Moreover, the local search considerably improves solutions obtained by the LR_MS method. From Table 6 we observe that CPLEX obtains equivalent gaps for instances with 6 items. From Table 7, we observe that the average gap is considerably improved for instances with high setup time factors.

Method **N**	**T**	**C**PLEX	**L**R_MS	**L**R_MS + LS
6	15	**2**	**1**	**3**
	30	**3**	0	**1**
12	15	0	0	**4**
	30	0	0	**4**
24	15	0	0	**4**
	30	0	0	**4**
48	15	**1**	**1**	**2**
	30	**1**	0	**3**
Total		**7**	**2**	**25**

Table 8: Computational results: Number of instances for which each method yields the best upper bound among the compared methods (1000 seconds)

In order to show that the LR_MS method with local search outperforms CPLEX, we show in Table 8 the number of instances where each method obtains the best upper bound. We use the same stopping criterion as before. The probing depth for LR_MS is fixed to 2 and the refining procedure is activated. The total number of instances where LR_MS outperforms CPLEX is 27, and CPLEX only outperforms LR_MS on 3 instances with 6 items and 2 instances with 48 items. Note also that, in two instances with 6 items, CPLEX and LR_MS obtain the same upper bound. The better results obtained by CPLEX on instances with 48 items can be explained by the probing depth of 2 that slow down LR_MS. When using a probing depth of 1 for instances with 48 items, we obtain better upper bounds.

Probing Depth		**0**		**1**		**2**		**3**	
N	T	gap	time	gap	time	gap	time	gap	time
6	15	23,7%	0,1	18,6%	0,9	14,5%	2,8	**13,9%**	4,8
	30	30,1%	0,3	24,5%	9,7	**18,9%**	27,9	18,9%	46,2
12	15	8,5%	0,2	7,0%	2,8	5,7%	10,0	**5,2%**	17,4
	30	12,6%	0,6	8,3%	29,3	**8,1%**	106,8	8,1%	186,3
24	15	3,8%	0,4	2,8%	9,7	**2,5%**	31,7	2,5%	51,9
	30	3,7%	1,2	3,0%	74,8	**2,4%**	367,6	2,3%	650,0
48	15	1,3%	0,7	0,9%	54,4	**0,8%**	189,0	0,8%	308,5
	30	2,7%	2,0	1,9%	418,8	**1,6%**	1711,3	1,6%	2884,4
Mean		10,8%	0,7	8,4%	75,0	**6,8%**	305,9	6,7%	518,7

Table 3: Impact of probing depth on the probing procedure.

Improving **p**rocedures		**L**R		**L**R + Refine		**L**R + Refine + LS	
N	T	**gap**	**time**	**gap**	**time**	**gap**	**time**
6	15	17,7%	2,1	14,5%	2,8	12,5%	3,7
	30	24,1%	22,8	18,9%	27,9	16,8%	43,7
12	15	6,8%	7,5	5,7%	10,0	4,5%	18,9
	30	9,7%	89,9	8,1%	106,8	5,6%	212,2
24	15	2,5%	24,1	2,5%	31,7	1,9%	158,5
	30	2,7%	291,6	2,4%	367,6	1,5%	1495,1
48	15	0,9%	139,0	0,8%	189,0	0,6%	810,1
	30	1,8%	1365,7	1,6%	1711,3	0,8%	8694,0
Mean		8,3%	242,8	6,8%	305,9	**5,5%**	1429,5

Table 4: Impact of the refining and local search procedures.

Method		**C**PLEX	**L**R_MS				**L**R_MS + LS			
Probing Depth		-	**0**	**1**	**2**	**3**	**0**	**1**	**2**	**3**
N	T									
6	15	2,9%	7,9%	6,3%	3,6%	3,2%	3,8%	4,9%	**2,8%**	3,0%
	30	11,1%	17,8%	14,7%	13,2%	13,2%	12,0%	10,4%	**11,0%**	**11,0%**
12	15	3,7%	4,7%	4,0%	3,5%	3,6%	3,4%	3,3%	3,3%	**3,1%**
	30	8,4%	8,3%	7,3%	7,0%	6,8%	5,0%	**4,9%**	5,1%	5,1%
24	15	2,1%	2,3%	1,9%	1,7%	1,7%	1,6%	**1,4%**	1,5%	1,5%
	30	3,4%	2,5%	2,4%	2,3%	2,3%	1,8%	1,8%	**1,6%**	1,7%
48	15	0,7%	0,8%	0,7%	0,6%	0,7%	0,6%	**0,5%**	0,6%	0,6%
	30	1,9%	1,5%	1,7%	1,9%	2,0%	1,3%	**1,2%**	1,4%	1,7%
Mean		4,3%	5,7%	4,9%	4,2%	4,2%	3,7%	3,6%	**3,4%**	3,5%

Table 6: Computational results: LR_MS vs. CPLEX (1000 seconds), impact of N and T.

Method	**C**PLEX	**L**R_MS				**L**R_MS + LS			
Probing Depth	-	**0**	**1**	**2**	**3**	**0**	**1**	**2**	**3**
Setup time factor									
1	3,3%	4,8%	4,3%	3,7%	3,5%	3,3%	3,3%	3,0%	**2,8%**
1,5	3,8%	5,4%	4,5%	3,7%	3,7%	3,3%	3,1%	**3,0%**	3,1%
2	4,6%	5,9%	5,0%	4,3%	4,2%	3,8%	3,9%	3,8%	**3,7%**
4	5,3%	6,6%	5,7%	5,2%	5,3%	4,3%	4,0%	**3,8%**	4,2%
Mean	4,3%	5,7%	4,9%	4,2%	4,2%	3,7%	3,6%	**3,4%**	3,5%

Table 7: Computational results: LR_MS vs. CPLEX (1000 seconds), impact of setup time factor.

6 CONCLUSION

In this paper, we addressed the multi-item capacitated lot-sizing problem with setup times and lost sales (MCLS-LS). To obtain good lower bounds, we used a Lagrangian relaxation of the capacity constraints. We also developed an algorithm that solves the induced single-item sub-problems optimally using an adaptation of the $O(T^2)$ dynamic programming algorithm of (Aksen et al., 2003). To find feasible solutions, we proposed an original approach based on a non-myopic heuristic that uses a probing strategy and a refining procedure. We also proposed some local search principles to improve the upper bound during the Lagrangian relaxation. Numerical results show that our methods outperform the CPLEX commercial solver. We obtain better or equivalent lower bounds in only few seconds. We also obtained better upper bounds when comparing CPLEX to a variant of the proposed method that is based on a multi-start strategy of the Lagrangian relaxation method. As a perspective to this work, it will be interesting to compare our approach to the resolution of the disaggregate formulation of MCLS-LS using CPLEX. It will also be interesting to use the Lagrangian lower bounds and upper bounds in a branch and bound approach.

REFERENCES

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

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

Absi N. and S. Kedad-Sidhoum, 2009. The multi-item capacitated lot-sizing problem with safety stocks and demand shortage costs. *Computers & Operations Research*, 36(11), p. 2926-2936.

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

Belvaux G. and L.A. Wolsey, 2000. bc-prod: A specialized branch-and-cut system for lot-sizing problems. *Management Science*, 46, p. 724-738.

Berk E., A.O. Toy, O. Hazir, 2008. Single item lot-sizing problem for a warm/cold process with immediate lost sales. *European Journal of Operational Research*, 187(3), p. 1251-1267.

Bitran G. and H.H. Yanasse, 1982. Computational complexity of the capacitated lot size problem. *Management Science*, 28, p. 1174-1186.

Brahimi N., S. Dauzère-Pérès, N.M. Najid, and A. Nordli, 2006. Single item lot sizing problems. *European Journal of Operational Research*, 168(1), p. 1-16.

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

Degraeve Z. and R. Jans, 2007. A New Dantzig-Wolfe Reformulation and Branch-and-Price Algorithm for the Capacitated Lot-Sizing Problem with Setup Times. *Operations Research*, 55(5), p. 909-920.

Fisher M.L., 1981. The Lagrangian relaxation method for solving integer programming problems. *Management Science*, 27, p. 1-18.

Held M., P. Wolfe, and H.D. Crowder, 1974. Validation of subgradient optimization. *Mathematical Programming*, 6, p. 62-88.

Jans R. and Z. Degraeve, 2007. Meta-heuristics for dynamic lot sizing: A review and comparison of solution approaches. *European Journal of Operational Research*, 177, p. 1855-1875.

Jans R. and Z. Degraeve, 2008. Modeling industrial lot sizing problems: a review. *International Journal of Production Research*, 46(6), p. 1619-1643.

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

Miller A.J., G.L. Nemhauser and M.W.P. Savelsbergh, 2003. On the polyhedral structure of a multi-item production planning model with setup times. *Mathematical Programming*, 94, p. 375-405.

Pochet Y. and L.A. Wolsey, 1991. Solving multi-item lot-sizing problems using strong cutting planes. *Management Science*, 37, p. 53-67.

Sandbothe R.A. and G.L. Thompson, 1990. A forward algorithm for the capacitated lot size model with stockouts. *Operations Research*, 38(3), p. 474-486.

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