Lagrangian heuristics for the capacitated multi-plant lot sizing problem with multiple periods and items

Desiree M. Carvalho¹, Mariá C. V. Nascimento¹,*

¹Instituto de Ciência e Tecnologia, Universidade Federal de São Paulo - UNIFESP
Av. Cesare G. Lattes, 1201, São José dos Campos, SP, Brazil

Abstract

Production planning plays an important role in the industrial sector. The focus of this paper is on the lot sizing of those companies composed by multiple plants, each of them with a finite planning horizon divided into periods. All plants produce the same items and have their demands to be met without delay. For producing items, all plants have a single machine with setup times and costs and a limited capacity of production. Transfers of production lots among plants and storage of items are allowed. Even though there are some studies to tackle this problem, to find feasible solutions for the entire set of benchmark instances remains a challenge. This paper introduces novel Lagrangian heuristics that, besides heuristically solving all benchmark instances, significantly outperformed the best heuristic from the literature.

Keywords: production, heuristics, lot sizing problem, multi-plant

1. Introduction

Production planning has been the subject of several researchers primarily due to the need for addressing the highly competitive productive sector. It is essential for a better use of resources and involves the decision making in a company for manufacturing and delivering products. The long-, medium- and short-term decisions are of utmost importance to optimize the costs related to

*Corresponding author. Tel.: +55 12 3309 9500
Email address: mcv.nascimento@unifesp.br (Mariá C. V. Nascimento)

May 26, 2016
the production process, from the acquisition of raw materials to final product delivery. Lot sizing, the problem focused in this paper, belongs to the short-term planning [5].

Accordingly, the multi-plant capacitated lot sizing problem (MPCLSP) aims at deciding in a finite planning horizon, the amount, when and what to produce to meet the demands of the plants. In addition, the demands must be satisfied without delay with the lowest cost possible. In particular, this study draws attention to the MPCLSP with multiple items and periods whose plants can produce any item, but have limited time (capacity) for operating the machines in each period. Moreover, in this problem, it is possible to keep inventory and to transfer production lots between plants. One can find in the literature a few studies on the MPCLSP [18, 19, 16]. Other variants of the lot sizing problem that are highly related to the MPCLSP can be found in [24, 12].

Nascimento et al. [16] generated a set of benchmark instances with different sizes and characteristics and put forward a solution method to heuristically solve the MPCLSP. It was not possible to find the optimal solutions for all instances by using the optimization package CPLEX v.7.5 because CPLEX ran out of either time or memory. Therefore, the authors proposed a hybrid metaheuristic that, however, could not find feasible solutions for a number of instances.

The primary goal of this paper is to find sharpened upper and lower bounds for the MPCLSP. In line with this, this paper includes as main contributions novel Lagrangian heuristics that significantly outperformed the state of the art heuristic for the MPCLSP, known as GPheur, in both experiments carried out. Additionally, taking the benchmark instances into account, comparing the achieved results with those obtained by CPLEX v. 12.6 [14] within a time limit of 1800 seconds, the proposed heuristics, named Lag and LaPRe, were very competitive, with better results for the high setup cost classes of instances. On average, LaPRe was twice better than CPLEX for these instances. Moreover, in an experiment with instances with a higher number of plants and items, Lag and LaPRe performed considerably better than CPLEX v. 12.6 and GPheur.

The remainder of this paper is organized as follows. Section 2 presents the
studied integer program. Section 3 loosely details the state of the art heuristic for solving the MPCLSP, and related works. Section 4 shows the proposed hybrid Lagrangian heuristic at length. Section 5 reports an analysis of the experiments carried out for attesting the effectiveness of Lag and LaPRe. To sum up, Section 6 summarizes the primary contributions of this paper and presents some final remarks.

2. The multi-plant, multi-item, multi-period capacitated lot sizing problem

This paper approaches the multi-period lot sizing problem with multiple plants, each of them with their amounts of items demanded in the periods (MPCLSP). Sambasivan and Schimidt [18] originally proposed the problem and presented a mathematical formulation, also discussed in [16]. Silva and Toledo [21] recently presented a reformulation of the MPCLSP, presented in formulation (1) - (6). This formulation is strongly related to the model found in [6].

Let $m$ be the number of plants, each of which indexed from 1 to $m$; $a$, the number of periods of the planning horizon, each of which indexed from 1 to $a$; and $n$, the number of items, each of which indexed from 1 to $n$. All demands are defined in advance as well as the production capacity of the plants in each period. Additionally, the MPCLSP allows transfers of production lots and the storage of production. For this reason, this study takes into account, for the production planning, the amount of an item $i$ to be produced in a plant-period $(j,t)$ to meet the demand of a plant-period $(k,u)$ for defining the corresponding variable $x_{ijtku}$.

Moreover, the following parameters must be explicit in the instances of this problem:
The setup cost for producing item $i$ at plant $j$;
the unitary production cost of item $i$ at plant $j$;
the unitary inventory cost of item $i$ at plant $j$;
the unitary transportation cost of items from plant $j$ to plant $k$;
the demand of item $i$ at plant $k$ in period $u$;
the processing time of item $i$ at plant $j$;
the setup time for preparing the machine for producing item $i$ at plant $j$;
the production capacity of plant $j$ in period $t$.

The mathematical formulation is presented next.

$$\min \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{t=1}^{a} \sum_{k=1}^{m} \sum_{u=1}^{a} (c_{ijtku} x_{ijtku}) + \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{t=1}^{a} (s_{ij} y_{ijt})$$  \hspace{1cm} (1)

subject to

$$\sum_{j=1}^{m} \sum_{t=1}^{a} x_{ijtku} = d_{iku} \hspace{1cm} \forall (i, k, u)$$  \hspace{1cm} (2)

$$\sum_{i=1}^{n} \left( f_{ij} y_{ijt} + \sum_{k=1}^{m} \sum_{u=t}^{a} b_{ij} x_{ijtku} \right) \leq P_{jt} \hspace{1cm} \forall (j, t)$$  \hspace{1cm} (3)

$$x_{ijtku} \leq \min \left\{ d_{iku}, \frac{P_{jt} - f_{ij}}{b_{ij}} \right\} y_{ijt} \hspace{1cm} \forall (i, j, t, k, u)$$  \hspace{1cm} (4)

$$x_{ijtku} \in \mathbb{N} \hspace{1cm} \forall (i, j, t, k, u)$$  \hspace{1cm} (5)

$$y_{ijt} \in \{0, 1\} \hspace{1cm} \forall (i, j, t)$$  \hspace{1cm} (6)

Let $\chi_{ijtku}$ be the costs of inventory and transfers between plants for producing item $i$ at plant $j$ in period $t$ to meet the demand at plant $k$ in period $u$. Bearing in mind that $r_{jj} = 0, \forall j \in \{1, \ldots, m\}$, Equation (7) shows a form to calculate these costs.

$$\chi_{ijtku} = \min_{1 \leq v \leq m} \{(u - t)e_{iv} + r_{jv} + r_{vk}\}$$  \hspace{1cm} (7)

Therefore, for calculating the corresponding costs for producing item $i$ at
plant \( j \) in period \( t \) to satisfy the demand \( d_{iku} \), one may use the following equation:

\[
\tau_{ijtku} = \begin{cases} 
0, & \text{if } u < t, \\
c_{ij} + \chi_{ijtku}, & \text{otherwise.}
\end{cases}
\]

According to cost \( \tau_{ijtku} \), once the variable \( x_{ijtku} \) is positive, item \( i \) will be stored from period \( u \) to \( t \) in the plant with the lowest inventory cost. The objective function, in Equation (1), aims at finding a solution with the lowest sum of the following costs: production, setup, inventory and transfers between plants. Constraints (2) force the production of the demands of all plants, whilst constraints (3) have as primary goal to keep production within the capacity limit of every plant in any of the periods. Constraints (4) limit the production at plant \( j \) in period \( t \) to meet the demand of item \( i \) of every pair plant-period \((k,u)\) taking the capacity \( P_{jt} \) into account. Finally, constraints (5) and (6) define, respectively, the domain of the variables \( x_{ijtku} \) to be natural values and \( y_{ijt} \) to be binary. Even though this formulation presents an asymptotically significant augment in the number of variables when comparing with the mathematical model introduced in [18], with regard to the instances introduced in [16], it was responsible for tighter linear relaxations [20].

3. Related Works

There are a few studies dealing with the MPCLSP in the literature. This specific problem was first investigated in [18]. The case study presented by the authors consists in the production planning of an American manufacturing company of steel rolls with plants positioned in different regions of the country. This type of problem concerns various productive sectors involving large companies, such as beverage corporations, mattress companies and chemical industries [1, 2, 8]. For heuristically solving the MPCLSP, Sambasivan and Yahya [19] presented a heuristic based on the Lagrangian relaxation of the capacity constraints of the mathematical formulation proposed in [18]. In the cases where the solutions of the approximate relaxed problem were infeasible for the MPCLSP, the
authors performed the same strategy used in [18] as an attempt to find feasible solutions.

Later, Nascimento et al. [16] developed a hybrid metaheuristic that significantly outperformed the aforementioned Lagrangian heuristic. This solution method is a Greedy Randomized Search Procedure (GRASP) embedded with the diversification strategy so-called path-relinking [10]. GRASP is a strategy with multiple iterations, each of which with two stages: a semi-greedy construction phase and a local search phase. The construction phase relies on the polynomial algorithm for the uncapacitated lot-sizing problem on parallel facilities [22]. The authors modified such algorithm for being semi-greedy for the uncapacitated multi-plant lot sizing problem. The authors introduced a feasibility strategy that shifts viable production lots among the plants and periods as an attempt to eliminate the capacity violations. If the resulting solution is feasible, then the local search phase starts. Otherwise, the iteration is over. Afterward, the path-relinking starts for then the iteration halts. As both their feasibility phase and local search were employed in the algorithms here proposed, they are thoroughly described in Sections 4.2.1 and 4.2.2.

In addition to the set of instances proposed by Sambasivan and Yahya [19], Nascimento et al. [16] generated larger and harder to solve instances. In spite of its good performance, the GRASP with path-relinking heuristic, named GPheur, could not solve all tested instances. Since then, no efficient strategy has been proposed, the primary reason that supported the proposal of the matheuristics introduced in this paper.

A number of studies in which authors explore relaxed problems to find upper bounds, a well-known branch of the matheuristics, to solve variants of the lot sizing problem have been investigated. Some of the most recent studies related to the investigated lot sizing problem are here mentioned.

Wu et al. [28] developed a relax-and-fix strategy to find upper bounds using information of the Lagrangian relaxation solutions for a lot sizing problem with multiple items and backorders. The Lagrangian relaxation is on the capacity constraints and the resulting problem can be solved by the classical polynomial
time algorithm that Wagner and Whitin [26] proposed. In the computational experiments, the authors analyse two different models of the same problem to employ the introduced strategy. They conclude that the model based on a facility location problem presented a better performance according to the studied set of instances.

A Lagrangian heuristic was recently proposed in [27] to approach an integrated lot sizing and fixed schedule problem. According to the authors, the problem can be interpreted as a variant of the classical CLSP. The Lagrangian relaxation is also performed on the capacity constraints. To determine upper bounds for the problem, they introduce a strategy based on lot shifts among periods applied to the Lagrangian solutions. Computational experiments were carried out and the results compared with those achieved by the optimal solver, XPRESS, with a time limit imposed. Although being very competitive, with large instances, XPRESS did not achieve better results than the Lagrangian strategy, that was very fast.

Lagrangian heuristics using the concept of hybridization with metaheuristic were proposed in [29] to solve an extension of the CLSP with carry-over. Also in the vein of matheuristics, Toledo et al. [23] proposed a strategy that integrates a genetic algorithm with a linear program to find approximate solutions for a bi-level lot sizing and scheduling problem. According to the computational results, the hybrid strategy had a competitive performance in comparison to strategies from the literature.

4. Solution Method

This section thoroughly discusses the proposed strategies to obtain both upper and lower bounds to the MPCLSP. Let one consider the MPCLSP relaxation of the capacity constraints. The introduced strategies employ the subgradient method to update the values of the Lagrange multipliers, here denoted by a matrix \( \lambda \) of dimensions \( m \times a \) [13]. To find the upper bounds for the problem, two heuristics are introduced: the first named Lagrangian heuristic (Lag)
and the second, Lagrangian heuristic with path-relinking (LaPRe). For ease of understanding, first, take into consideration the following Lagrangian model.

\[
\min \sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij} \sum_{t=1}^{m} \sum_{k=1}^{a} \sum_{u=1}^{a} c_{ijtku} x_{ijtku} + \sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij} \sum_{t=1}^{m} \sum_{k=1}^{a} \sum_{u=1}^{a} s_{ij} y_{ijt} - \sum_{j=1}^{m} \sum_{t=1}^{a} \lambda_{jt} \left( \sum_{i=1}^{n} f_{ij} y_{ijt} + \sum_{k=1}^{m} \sum_{u=t}^{a} b_{ij} x_{ijtku} \right) - P_{jt} \right) \tag{8}
\]

subject to:

\[
\sum_{j=1}^{m} \sum_{t=1}^{a} x_{ijtku} = d_{iku} \quad \forall (i, k, u) \tag{9}
\]

\[
x_{ijtku} \leq \min \left\{ d_{iku}, \frac{P_{jt} - f_{ij}}{b_{ij}} \right\} y_{ijt} \quad \forall (i, j, t, k, u) \tag{10}
\]

\[
x_{ijtku} \in \mathbb{N} \quad \forall (i, j, t, k, u) \tag{11}
\]

\[
y_{ijt} \in \{0, 1\} \quad \forall (i, j, t) \tag{12}
\]

To the best knowledge of the authors, there is no polynomial algorithm for solving the resulting relaxed problem. For this reason, the tool employed to solve the relaxed problem was CPLEX v. 12.6 \[14\]. CPLEX is a widely used mixed integer programming optimization tool efficient in finding exact/approximate solutions to combinatorial optimization problems. In preliminary experiments, CPLEX experienced difficulties in solving instances of the relaxed model (8)-(12) within reasonable CPU time. That is the reason why the chosen strategy decomposes it per item, subdividing the MPCLSP into \( n \) independent subproblems. Each relaxed subproblem, therefore, results in the following formulation, for \( 1 \leq i \leq n \).
\[
\min L_i(\lambda) = \sum_{j=1}^{m} \sum_{t=1}^{a} \sum_{k=1}^{m} \sum_{u=1}^{a} (\bar{r}_{ijtku} x_{ijtku}) + \sum_{j=1}^{m} \sum_{t=1}^{a} (s_{ij} y_{ijt}) \\
- \sum_{j=1}^{m} \sum_{t=1}^{a} \lambda_{jt} \left( f_{ij} y_{ijt} + \sum_{k=1}^{m} \sum_{u=t}^{a} b_{ij} x_{ijtku} \right) - P_{jt}
\] 

(13)

subject to:

\[
\sum_{j=1}^{m} \sum_{t=1}^{a} x_{ijtku} = d_{iku} \quad \forall (k, u)
\] 

(14)

\[
x_{ijtku} \leq \min \left\{ d_{iku}; \left\lfloor \frac{P_{jt} - f_{ij}}{b_{ij}} \right\rfloor \right\} \quad \forall (j, t, k, u)
\] 

(15)

\[
x_{ijtku} \in \mathbb{N} \quad \forall (j, t, k, u)
\] 

(16)

\[
y_{ijt} \in \{0, 1\} \quad \forall (j, t)
\] 

(17)

The Lagrangian function of problem (1)-(6) is then \(\sum_{i=1}^{n} L_i(\lambda)\). It is well known from Lagrangian duality theory that, in this case, the best lower bound is the optimal solution of the dual problem:

\[
\max_{\lambda \geq 0} \left\{ \sum_{i=1}^{n} \min_{\lambda} L_i(\lambda) : \text{s.a. (14) \& (17)} \right\}
\] 

(18)

So, the problem of determining the best choice for \(\lambda\) has to be addressed. To perform this task, Fisher [9] underlines three main approaches in his study: the subgradient method; versions of the simplex method with column generation; and multiplier adjustment methods. The second option has shown to be too expensive in comparison to the subgradient method. The multiplier adjustment methods are virtually heuristics specifically tailored for some problem as that proposed in [7]. Even though these adjustment strategies have a high potential, outperforming the subgradient method in some case studies, the general nature subgradient method and its robustness make it the most used tool for determining the Lagrange multipliers. Additionally, the good results found in [13, 25, 24] and the low complexity of the subgradient method motivated us to
employ this strategy. The version of the subgradient method used in this study is explained in the next section.

4.1. Subgradient Method

The subgradient algorithm introduced in [13] has been widely employed in the literature [3]. It iteratively updates the Lagrangian multipliers according to a sequence of scalars known as step sizes, as in Equation (19).

$$\lambda_{jt}^{v+1} = \max\{0, \lambda_{jt}^v + \alpha^v g_{jt}^v(x^v, y^v)\} \quad \forall (j, t)$$ (19)

where $\alpha^v$ is the step size of the method in iteration $v$; $(x^v, y^v)$ corresponds to the values of the decision variables in iteration $v$; and $g_{jt}^v(x^v, y^v)$ is the difference between the time required for producing all items in period $t$ at plant $j$ according to the relaxed solution in iteration $v$ and the capacity limit in pair plant-period $(j, t)$, calculated as in Equation (20).

$$g_{jt}^v(x^v, y^v) = \sum_{i=1}^{n} f_{ij}y_{ijt} + \sum_{k=1}^{m} \sum_{u=t}^{a} b_{ij} x_{ijtku} - P_{jt} \quad \forall (j, t)$$ (20)

It is noteworthy the need for initial values for parameters $\alpha^v$ and $\lambda_{jt}^v$, i.e., $\alpha^0$ and $\lambda_{jt}^0$, for the iterative method. For a better organization of this paper, Appendix A reports all initial values, chosen values of the parameters and experiments for parameter tuning.

The choice of the step size, $\alpha^v$, is of utmost importance for the convergence of the subgradient method to the optimal solution. Thus, the updating rule used for $\alpha^v$ was suggested in [13], indicated in Equation (21).

$$\alpha^v = \sigma^v \left( \frac{Z_{sup}^v - Z_{inf}^v}{\|g_{jt}^v(x^v, y^v)\|^2} \right)$$ (21)

In Equation (21), $Z_{sup}^v$ and $Z_{inf}^v$ are, respectively, the upper and lower bound
solutions found up to iteration \( v \). It is expected that the step size \( \alpha^v \to 0 \) to ensure the convergence of the method. As in Equation (20), \( \alpha^v \) depends on an approximate solution \((Z^v_{sup})\). If no feasible \( Z^v_{sup} \) has been found up to iteration \( v \), a large number was employed in order to estimate such value. To guarantee that \( \alpha^v \to 0 \), one must consider a decreasing sequence \( \sigma^v \) that tends to zero [13]. Since \( \sigma^v \) is responsible for regulating the convergence rate of the subgradient method, as suggested in [13], a good rule is to update \( \sigma^v \) dividing it by a factor \( \eta \) after a number of iterations, \( q_t \). The fine tuning of \( \sigma^0 \) and of the reduction factor \( \eta \) is presented in Appendix A. Additionally, it presents a restarting mechanism that consists in updating \( \alpha^v \) after a number of iterations that a solution remains unchanged.

Since the convergence of the method may require a high CPU running time, in most of the cases, the method stops when it reaches the running time limit. Additionally, if the introduced Lagrangian heuristic provides an upper bound that coincides with the lower bound, the method halts and returns the optimal solution for the primal problem. Next section shows the proposed Lagrangian heuristics.

4.2. Lagrangian Heuristic

To obtain the upper bounds, this paper proposes a Lagrangian heuristic that applies a feasibility stage to each solution \((x^v, y^v)\) of the Lagrangian relaxation problem. The following steps summarize the process that derives the upper bounds of the introduced strategy.

- **Definition of the lower bound**

  Let \( \lambda^v = \{\lambda^v_{jt} \geq 0: j \in \{1, 2, \ldots, m\} \text{ and } t \in \{1, 2, \ldots, a\}\} \) be the set of Lagrange multipliers associated with the capacity constraints in iteration \( v \). Determine \((x^v, y^v)\) by solving the relaxed problems:

  \[
  Z_{iLR}^v(\lambda^v) = \min L_i(\lambda^v) : s.t. (14) - (17) \quad \forall i = \{1, \ldots, n\}
  \]  

(22)
Since problems (22) are $\mathcal{NP}$-hard [17], the well-known mixed-integer program solver CPLEX version 12.6 [14] was chosen to solve them. Therefore, the solution value for the relaxed problem is $Z_{LR}^v(\lambda^v) = \sum_{t=1}^n Z_{LR}^v(\lambda^v)$. Once one has the relaxed solution, one must verify whether or not to update the largest lower bound found up to that iteration.

- **Definition of a Feasible Solution (upper bound)**

If the Lagrangian solution found is feasible for the capacitated problem as well, it is enough to update the upper bound ($Z_{\text{sup}}^v$) with the relaxed solution value, $Z_{LR}^v$, and to return the optimal primal solution. However, if the solution is not feasible for the capacitated problem, the variables of the relaxed solution go through a feasibility process. The feasibility strategy proposed in this paper hybridizes two different strategies. The first, proposed in [16], virtually transfers production lots amongst periods. The second, based on that found in [24], was originally proposed to tackle the parallel machines capacitated lot sizing problem and was suitably adapted to deal with the MPCLSP.

4.2.1. Feasibility Phase I

Maes et al. [15] demonstrated that to find feasible solutions for the capacitated lot sizing problem is $\mathcal{NP}$-complete. Therefore, the proposed Lagrangian heuristics include the feasibility strategy set out in [16] to find feasible solutions from the approximations of the dual Lagrangian problem. It consists in a local search-based strategy, here named Feasibility Phase I.

For simplicity, let $\Delta P(j, t) = \sum_{l=1}^n (f_{lj} y_{ljt} + \sum_{k=1}^m \sum_{a=t}^a h_{ljx_{ljtku}}) - P_{jt}$ be the total time required at plant $j$ in period $t$ due to the approximate dual solution found in a given iteration of the Lagrangian strategy. For both enhancing the solution quality and either diminishing or even eliminating the possible overtime of the capacity constraints, Feasibility Phase I shifts production lots among periods and plants with movements denoted by MOVE1 and MOVE2, explained next.
• **MOVE1** - This move assesses the costs with regard to the transfer of a certain amount of item $i$, produced at plant $j$ in period $t$, to a plant $jd$, $1 \leq jd \leq m$, in period $td$, $1 \leq td < t$ or in period $td = t$, for all $j \neq jd$. The tuple (quantity, item, origin plant, origin period, destiny plant, destiny period) that provides the largest reduction in the solution costs is then chosen. The analysed options to define the quantity of items to be transferred are:

1. the total quantity of item $i$ produced at plant $j$ in period $t$, as presented in Equation (23):
   \[ q_1 = \sum_{k=1}^{m} \sum_{u=t}^{a} x_{ijtku} \]  
   (23)

2. the minimum between the amount produced of item $i$ at plant $j$ in period $t$ and the quantity of item $i$ that, if not produced in the pair $(j,t)$, eliminates the overuse of the capacity of plant $j$ in period $t$, $OP_{jt}$:
   \[ q_2 = \min \{ q_1, OP_{jt} \} \]  
   (24)
   where
   \[ OP_{jt} = \max \{ 0, \lceil (\Delta P(j, t)) / b_{ij} \rceil \} \]

3. the minimum between the amount produced of item $i$ in plant-period $(j, t)$ and the quantity of the same item for production in plant-period $(jd, td)$ without exceeding the pair $(jd, td)$, $AP_{jd,td}$:
   \[ q_3 = \min \{ q_1, AP_{jd,td} \} \]  
   (25)
   where
   \[ AP_{jd,td} = \max \{ 0, \lfloor (f_{i,jd}(y_{ijd,td} - 1) - \Delta P(jd, td)) / b_{i,jd} \rfloor \} \]

• **MOVE2** - This move aims at transferring partially or the entire produc-
tion lots of an item \( i \) from some plant \( j \) in a period \( t \) to future periods \( td, t + 1 \leq td \leq a \). For this movement, it is necessary to calculate the inventory of an item \( i \) stored in plant \( j \) up to period \( t \) to meet demands of other plants and later periods, as indicated in Equation (26):

\[
I_{ijt} = \sum_{b=1}^{m} \sum_{u=1}^{t} \sum_{k=1}^{m} \sum_{s=t+1}^{a} \beta x_{ibuks}
\]

where

\[
\beta = \begin{cases} 
1, & \text{if } \arg \min \chi_{ibuks} = j \\
0, & \text{otherwise}
\end{cases}
\]

Similar to MOVE1, let one consider a total of three options of transfer lots for deciding the best option:

1. the minimum amount of inventory among periods from \( t \) to \( td - 1 \) of item \( i \) produced at plant \( j \) \((mn = \min_{t' \in \{t,...,td-1\}} I_{ijt'})\) and the total produced of \( i \) in pair plant-period \( (j,t)\):

\[
q4 = \min \{q1, mn\}
\]

2. the minimum between \( mn = \min_{t' \in \{t,...,td-1\}} I_{ijt'} \) and the quantity of item \( i \) that eliminates the overtime capacity at plant \( j \) in period \( t \):

\[
q5 = \min \{mn, OP_{jt}\}
\]

3. the minimum between \( mn = \min_{t' \in \{t,...,td-1\}} I_{ijt'} \), the total produced of \( i \) in pair plant-period \( (j,t) \) and the quantity of the same item for production in plant-period \( (jd,td) \) without exceeding the pair \( (jd,td)\):

\[
q6 = \min \{mn, q1, AP_{jd,td}\}
\]

For calculating the costs of infeasible solutions, similar to Gopalakrishnan et al. [11], this feasibility procedure embeds the overtime capacity into the ob-
jective function as a penalty. This penalty benefits movements that reduce the overtime capacity, thereby reducing the infeasibility of the solution. Therefore, if in the end of the transfer process the solution is infeasible, the transfer procedure returns the solution value, $OF'$, described in Equation (30).

$$OF'(x, y) = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{a} \sum_{t=1}^{m} \sum_{u=1}^{a} (c_{ijtku}x_{ijtku}) + \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{t=1}^{a} (s_{ijyt}) + \rho \times over \_P \tag{30}$$

where $\rho$ is the rate to penalize the capacity constraints, empirically fit (for more, see Appendix A); and $over \_P$ is equal to the overall overtime capacity. Algorithm 1 displays a pseudocode of this feasibility strategy, Feasibility Phase I.

**Algorithm 1: Feasibility Phase I.**

**Data:** Approximate dual solution $(x^v, y^v)$

**Result:** Either a heuristic solution for the primal problem or an infeasible solution

1. repeat
2. Search for the pair plant-period with the largest overtime: $(j, t)$;
3. Calculate $OF'(x^v, y^v)$ as in Equation (30);
4. $(x^v_{sup}, y^v_{sup}) \leftarrow (x^v, y^v)$;
5. $best \_saving \leftarrow 0$;
6. forall the $td, jd, i$ do
7. if there exists production of item $i$ in pair plant-period $(j, t)$ then
8. if $k \neq j$ and $t = td$ or $t > td$ then
9. Evaluate MOVE1 if the best tuple $(q, i, j, t, jd, td)$ provides a better saving than $best \_saving$;
10. else
11. if $t < td$ then
12. Evaluate MOVE2 if the best tuple $(q, i, j, t, jd, td)$ provides a better saving than $best \_saving$;
13. Perform the move corresponding to $best \_saving$, if it is positive;
14. Update $sol$ subtracting $best \_saving$ to it;
15. if $(x^v_{sup}, y^v_{sup})$ is feasible then
16. return $(x^v_{sup}, y^v_{sup})$ and sol
17. until the stop criterion has not been reached;

In Algorithm 1, the stop criterion is either a number of iterations (value
reported in Appendix A) or the solution achieved is a local optimum.

4.2.2. Local Search

The local search here employed is the same introduced in [16]. It consists in choosing in a greedy fashion the movements of the Feasibility Phase I method, explained in Section 4.2.1. However, it just takes into account those tuples that keep the solution feasible.

4.2.3. Feasibility Phase II

Similar to the Feasibility Phase I strategy, Feasibility Phase II searches for viable solutions by transferring production batches to different periods and plants. It has been adapted from the original version to solve the capacitated lot sizing problem with parallel machines [24]. For this, first, consider again

\[
\Delta P(j,t) = \sum_{i=1}^{n} b_{ij} \sum_{k=1}^{m} \sum_{u=t}^{a} x_{ijaktu} + f_{ijy_{jlt}} - P_{jt}.
\]

To transfer an amount \( q \) of item \( i \) from pair plant-period \((j,t)\) to \((jd,td)\), one must evaluate the resulting overtime capacity of pairs plant-period \((j,t)\) and \((jd,td)\). Accordingly, the overtime capacity in pair \((jd,td)\) before any transfer of item \( i \) is:

\[
Cap_{\text{Before}}(jd,td) = \max\{0, \Delta P(jd,td)\}
\]

After the transfer, the overtime capacity of plant-period \((j,t)\) will be:

\[
Cap_{\text{After}}(q,i,j,t) = \max\{0, \Delta P(j,t) - qb_{ij} - f_{ij}\delta_1\}
\]

where

\[
\delta_1 = \begin{cases} 
1, & \text{if } q = \sum_{k=1}^{m} \sum_{u=t}^{a} x_{ijaktu} \\
0, & \text{if } q < \sum_{k=1}^{m} \sum_{u=t}^{a} x_{ijaktu}
\end{cases}
\]

Moreover, the overtime capacity in pair \((jd,td)\) after transferring \( q \) units of item \( i \) is:

\[
Cap_{\text{After}}(q,i,jd,td) = \max\{0, \Delta P(jd,td) + qb_{i,jd} + f_{i,jd}\delta_2\}
\]
\[
\delta_2 = \begin{cases} 
1, & \text{if } \sum_{k=1}^{m} \sum_{u=t}^{a} x_{ijdtku} = 0 \\
0, & \text{if } \sum_{k=1}^{m} \sum_{u=t}^{a} x_{ijdtku} > 0 
\end{cases}
\]

Therefore, to perform Feasibility Phase II, it is needed to define from a given plant-period \((j, t)\) with overtime capacity, which item \(i\) and its amount \(q\) to transfer and to which pair plant-period \((jd, td)\). For such, it first considers the transfers when \(t \geq td\), characterizing the backward stage. In the backward stage, lots of items \(i\), \(1 \leq i \leq n\), are transferred from \((j, t)\), \(\forall j\) and \(2 \leq t \leq a\), to a pair plant-period \((jd, td)\), \(\forall jd\) and \(1 \leq td \leq t\), to eliminate the overtime capacity in \((j, t)\), just disregarding \((jd, td) = (j, t)\). The origin of the transfer is chosen by investigating in each period, starting in \(p\) to 2, all \(j\)’s with overtime capacity to form the pair \((j, t)\). Therefore, the aim is to investigate the candidates to form the pair \((j, t)\) following the sequence: \((1, a)\), \((2, a)\), \ldots, \((m, a), (1, a - 1), (2, a - 1), \ldots, (m, 2)\). Bearing this in mind, the criterion employed to choose for each \((j, t)\), the tuple \((i, q, jd, td)\) is:

\[
\min_{\forall (q, i, jd, td)} \frac{\text{Penalty}(q, i, j, t, jd, td)}{\text{Overtime Reduction}(q, i, j, t)} (32)
\]

subject to:

\[
q = \min \left\{ \sum_{k=1}^{m} \sum_{u=t}^{a} x_{ijtku} \Delta P(j, t)/b_{ij} \right\} \quad \forall t \geq td, i
\] (33)

where \(\text{Overtime Reduction}(j, t)\) is the overtime capacity reduction in plant-period \((j, t)\) with the transfer, calculated as:

\[
\text{Overtime Reduction} = qb_{ij} + f_{ij} \delta_1
\] (34)

and \(\text{Penalty}(q, i, j, t, jd, td)\) consists in the sum of the overtime capacity in plant-period \((j, t)\) and \((jd, td)\) after the lot transfer. Accordingly, \(\text{Penalty}(q, i, j, t, jd, td)\) can be expressed as in Equation (35).
\[ \text{Penalty}(q, i, j, t, jd, td) = \text{Cap\_After}(q, i, j, t) + [\text{Cap\_After}(q, i, jd, td) - \text{Cap\_Before}(jd, td)] \]  

(35)

Let one define a round of the backward stage as when the problem (32)-(33) has been solved for every pair \((j, t)\) with overtime capacity. It is worth mentioning that it is very likely that such routine leads to an overtime in the first period. On the one hand, if all pairs \((j, 1), 1 \leq j \leq m\), have no overtime capacity, then the method returns this feasible solution. On the other hand, if the solution returned is not feasible, the forward stage starts.

The forward step relies on distributing the overtime capacity in plant-period \((j, t), 1 \leq t \leq a - 1\), by transferring production lots to some pair plant-period \((jd, td), \forall jd\) and \(t \leq td \leq a\), just disregarding \((jd, td) = (j, t)\). At each period \(t\), from 1 to \(a - 1\), the forward phase searches for the plants \(j\) with overtime capacity, following the increasing order of the plant index. Therefore, for each \((j, t)\), the criterion employed for choosing the tuple \((i, q, jd, td)\) also aims at the objective function in Equation (32), but subject to the following constraints:

\[ q = \min \{ \sum_{k=1}^{m} \sum_{u=td}^{a} x_{ijk} \Delta P(j, t)/b_{ij} \} \quad \forall t \leq td, i \]  

(36)

It is worth mentioning that unlike the backward phase, this stage may not necessarily eliminate the overtime capacity of a given pair plant-period \((j, t)\).

This phase explores all possible \((j, t)\) until either the current solution is feasible or all pairs \((j, t)\) have been investigated. Feasibility Phase II stops if the returned solution is feasible. Otherwise, it executes the backward stage, followed by the forward stage until either a feasible solution is found or the number of times both stages have run reached a limit, the same stop criterion proposed in [24]. Algorithm 2 summarizes the steps of this strategy.
Algorithm 2: Feasibility Phase II.

**Data:** Approximate dual solution \((x^{v, sup}_v, y^{v, sup}_v)\)

**Result:** Either a heuristic solution for the primal problem or an infeasible solution

1. repeat
   2. /* Backward Phase */
   3. for \(t = a\) to 2 with overtime capacity do
      4. repeat
         5. Identify the first plant \(j\) with positive overtime capacity in period \(t\);
         6. repeat
            7. Choose the best tuple \((q, i, j, t, jd, td)\) and update \((x^{v, sup}_v, y^{v, sup}_v)\) by performing the corresponding transfer if \(q > 0\);
            8. until pair \((j, t)\) does not present overtime capacity;
            9. until \(j = \emptyset\);
      10. if \((x^{v, sup}_v, y^{v, sup}_v)\) is feasible then
          11. return \((x^{v, sup}_v, y^{v, sup}_v)\)
          12. else
             /* Forward Phase */
             13. for \(t = 1\) to \(a - 1\) with overtime capacity do
                14. Mark \(j = 1, \ldots, m\);
                15. repeat
                   16. Identify the first marked plant \(j\) with positive overtime capacity in period \(t\);
                   17. Choose the best tuple \((q, i, j, t, jd, td)\) and update \((x^{v, sup}_v, y^{v, sup}_v)\) by performing the corresponding transfer if \(q > 0\);
                   18. if \(q = 0\) then
                      19. Unmark plant \(j\);
                   20. until \(j = \emptyset\) or all \(j\)'s are unmarked;
                   21. if \((x^{v, sup}_v, y^{v, sup}_v)\) is feasible then
                      22. return \((x^{v, sup}_v, y^{v, sup}_v)\)
                   23. else
                      24. until the stop criterion has not been reached;

4.3. Path-relinking phase

Path-relinking (PR) is a heuristic usually blended with diversification-based strategies for better exploring the search space by using solutions found by heuristic methods. The principle underlying PR is that it enables the intensification and diversification during the search process, since it may encounter high
quality solutions that belong to paths between pairs of solutions.

Like in [16], that uses PR embedded to GRASP, in this paper, the proposed Lagrangian heuristic is hybridized with PR to enhance the quality of the solutions found by the strategy. First, an attempt to embed the same PR version as in [16] into the Lagrangian heuristic improved the results achieved by the pure heuristic. However, it neither outperformed CPLEX within the imposed time limit in most of the cases nor was very competitive for a set of instances proposed in [16]. Then, motivated by the high performance of the current version of CPLEX, a novel PR was developed.

In this strategy, all feasible solutions obtained by the Lagrangian heuristic become one of the extremes of the path (known as *guiding* and *initial solutions*). Let $S_{\text{initial}}$ be the set of tuples $(i,j,t)$ such that the decision variables $y_{ijt}$ from the current feasible solution is equal to 1. Consider also $S_{\text{guiding}}$ the set of tuples $(i,j,t)$ such that the decision variables $y_{ijt}$ is equal to 1 in the best solution found so far. To find the best solution by combining elements from both solutions, it is enough to solve the problem (8)-(12), with the following additional constraints:

$$y_{ijt} = 0 \quad \forall i, j, t \notin \{S_{\text{initial}}, S_{\text{guiding}}\} \quad (37)$$

To sum up, Figure 1 displays a flowchart of LaPRe. For the version of the Lagrangian heuristic without the path-relinking, just disregard the PR phase.

The next section presents the computational experiments carried out for attesting the quality of the introduced solution methods.

5. Computational Experiments

This section shows the results of two experiments performed on a number of instances to evaluate the quality of the results obtained by the Lagrangian heuristics. The first experiment assesses the effectiveness of the heuristics here proposed by comparing their results with those achieved by the optimization solver CPLEX v. 12.6 [14] and with the state of the art heuristic for the
Read input
Set Lagrangian initial variables $\lambda, \pi$
Solve the Lagrangian problem $Z_n \leftarrow \Sigma \min L_i(\lambda)$
Update subgradient variables $\lambda, g, \alpha$
Update the best Lagrangian lower bound $Z_0$
Has the stop criterion been reached?
yes
no
Update subgradient variables $\lambda, g, \alpha$
Apply the local search to the solution and update the overall solution
Is the solution feasible?
yes
no
Apply the local search to the solution and update the overall solution
Apply the Feasibility II to the corresponding solution
yes
no
Apply Feasibility I to the corresponding $Z_n$
Is the solution feasible?
yes
no
Stop
Start

Figure 1: This flowchart presents the main steps of LaPRe.
MPCLSP, GPheur [16]. In the second experiment that further evaluates the Lagrangian heuristics, their results on large-sized instances are compared with those obtained by GPheur and CPLEX v.12.6. To ensure a fair comparison in the computational experiments, the stop criterion considered for all solution methods in the experiments was the time limit of 1800 seconds for each instance.

All computational tests were carried out in a cluster with 104 computer nodes, each node with two Intel Xeon E5-2680v2 ten-core processor of 2.8 GHz, 128 GB DDR3 RAM. These experiments used 8 threads, and the implementations of the algorithms were in C language. The first experiment was conducted using an 8-class ensemble of 480 instances, set out in [16]. The choice for the parameter values of these instances was in accordance with a uniform distribution within a continuous interval, as indicated next.

- Unitary cost of production: $c_{ijt} \sim U[1, 5; 2, 5]$
- Unitary inventory cost: $e_{ij} \sim U[0, 2; 0, 4]$
- Unitary transfer cost: $r_{jk} \sim U[0, 2; 0, 4]$
- Unitary production time: $b_{ij} \sim U[1, 0; 5, 0]$
- Demand: $d_{iku} \sim U[0, 180]$

Accordingly, the type of capacity, setup cost and setup time of the instances are the underlying features that characterize their division into classes.

- Type of Capacity - Tight (T) or Normal (N)
  - Let one consider $P_{jt} = \left[ \sum_{t=1}^{a} \sum_{i=1}^{n} \left( \frac{d_{iku} b_{ij} + f_{ij}}{a} \right) \right]$, when the instance has “normal” capacity. For the instances with “tight” capacity, consider $P_{jt} = 0.9 \left[ \sum_{t=1}^{a} \sum_{i=1}^{n} \left( \frac{d_{iku} b_{ij} + f_{ij}}{a} \right) \right]$.

- Setup cost - High (H) or Low (L)
  - For generating instances with low setup cost, Nascimento et al. [16] suggested to pick a value at random for every $s_{ijt}$ within the interval
Concerning high setup cost instances, besides assigning random values from the same interval for every \( s_{ijt} \), it is supposed to multiply the resulting value by 10.

- Setup time - High (H) or Low (L)
  - One the one hand, for generating instances with low setup time, it suffices to consider \( \forall f_{ij} \) a value picked at random from the interval [10.0; 50.0]. On the other hand, for producing high setup time instances, Nascimento et al. [16] recommended, in addition to the previous strategy for assigning values for every \( f_{ij} \), to multiply the resulting value by 1.5.

Therefore, an instance from class NHL has normal capacity, high setup costs and low setup times. In the first set of instances, generated in [16], each instance corresponds to a problem with 12 periods, 2, 4 or 6 plants and 6, 12, 25 or 50 items. For each combination, the set has 5 problems, totaling 60 instances in each class.

By using the same methodology for generating instances, this paper proposes large instances for composing the set of instances of the second experiment. The chosen configuration for composing the pool of instances is 12 periods, 15 or 20 plants and 70, 80 or 90 items. Similar to the first set of instances, each combination of the configurations provided 5 problems, totalling 240 instances, 30 in each class.

To estimate how far the solutions are from the optimal solution, one may calculate the gap as follows.

\[
gap = 100 \times \frac{(Z_{sup} - Z_{inf})}{Z_{inf}}\%
\]

where \( Z_{sup} \) is the upper bound solution achieved by some method and \( Z_{inf} \) is a lower bound that depends on the experiment.

In addition to the average results achieved by the three algorithms, to enable a more detailed performance analysis of the algorithms, both experiments assess
the performance profiles of Dolan and Moré [4]. For this, consider \( \zeta \) and \( \Psi \) to be, respectively, the set of \( n_\zeta \) algorithms and the set of \( n_\psi \) instances taken in the experiments. The \emph{performance ratio} of an algorithm \( \zeta \in \zeta \) with regard to an instance \( \psi \in \Psi \), here denoted as \( o_{\zeta \psi} \), is calculated according to Equation (38).

\[
o_{\zeta \psi} = \frac{t_{\zeta \psi}}{\min\{t_{\zeta \psi'} | \zeta' \in \zeta\}} \tag{38}
\]

In Equation (38), let \( t_{\zeta \psi} \) be the value of the metric to be analysed in algorithm \( \zeta \) with \( \psi \) as input. In the experiments, both solution values and time to solutions will be under investigation. According to Dolan and Moré [4], in the original form of the metric, the lower \( t_{\zeta \psi} \) is, the better. It is noteworthy that, according to Equation (38), the best \( o_{\zeta \psi} \) value is 1, that occurs when \( t_{\zeta \psi} \) value is the minimum among all algorithms from set \( \zeta \).

Moreover, in this profile, the \( y \)-axis indicates the performance ratio. It is given by the number of problems that a certain algorithm achieved a \emph{performance ratio} equal to or better than a coefficient \( \tau \), the \( x \)-axis of the graph, divided by the cardinality of \( \zeta \). In other words, \( \Theta(\zeta)(\tau) = \frac{1}{n_\psi} | \psi \in \Psi : \log_2(o_{\zeta \psi}) \leq \tau \| \). Therefore, \( \Theta(\zeta)(\tau) \) indicates the probability of \( \zeta \) having a \emph{performance ratio} within a factor \( \tau \).

The following sections present the results obtained by the proposed Lagrangian heuristics, CPLEX v. 12.6 and GRASP heuristic with path-relinking [16].

5.1. Experiment I

The lower bound employed for calculating the gaps was the best between that provided by CPLEX, the Lagrangian heuristics and the linear relaxation of the model introduced in [20]. The first part of the experiment contrasts the results obtained by the Lagrangian heuristics, considering the non-hybridized version, identified as Lag, and the version with path-relinking (LaPRe), to those reached

\[\text{The best lower bounds are available in } \text{https://sites.google.com/site/nascimentomcv/downloads/mpclsp.}\]
by CPLEX v. 12.6. Between the two existing formulations, the CPLEX solver achieved its best results for the instances of this experiment using formulation (1)-(6).

In Table 1, the columns labeled as CPLEX, LaPRe and Lag indicate the average results achieved by these algorithms for each pair of number of plants and of items (all instances have 12 periods). The averages are with regard to the gaps, AG (%), achieved by the algorithms and the time to achieve the best solution, AT (sec).

Table 1: Results obtained by CPLEX, the Lagrangian heuristic with path-relinking (LaPRe) and the Lagrangian heuristic without path-relinking (Lag) for each type of instances. For each type of instances, this table reports the average gaps obtained with the CPLEX lower bounds (AG) and the average times (AT), in seconds, to find the best solutions.

<table>
<thead>
<tr>
<th>Plants</th>
<th>Items</th>
<th>CPLEX</th>
<th>LaPRe</th>
<th>Lag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>AG (%)</td>
<td>AT (sec)</td>
<td>AG (%)</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0.01</td>
<td>432.80</td>
<td>0.19</td>
</tr>
<tr>
<td>12</td>
<td>0.04</td>
<td>528.93</td>
<td>1066.20</td>
<td>1.53</td>
</tr>
<tr>
<td>25</td>
<td>0.14</td>
<td>1241.12</td>
<td>1256.88</td>
<td>2.55</td>
</tr>
<tr>
<td>50</td>
<td>0.07</td>
<td>1193.40</td>
<td>1088.33</td>
<td>3.48</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>0.37</td>
<td>909.50</td>
<td>0.49</td>
</tr>
<tr>
<td>12</td>
<td>0.54</td>
<td>1427.30</td>
<td>1234.08</td>
<td>2.07</td>
</tr>
<tr>
<td>25</td>
<td>0.44</td>
<td>1466.45</td>
<td>1223.83</td>
<td>1.56</td>
</tr>
<tr>
<td>50</td>
<td>0.25</td>
<td>1299.80</td>
<td>1102.13</td>
<td>1.75</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>1.65</td>
<td>1299.70</td>
<td>1.37</td>
</tr>
<tr>
<td>12</td>
<td>0.87</td>
<td>1463.40</td>
<td>1349.43</td>
<td>3.47</td>
</tr>
<tr>
<td>25</td>
<td>3.17</td>
<td>1596.33</td>
<td>1166.23</td>
<td>1.71</td>
</tr>
<tr>
<td>50</td>
<td>7.86</td>
<td>1539.88</td>
<td>1186.40</td>
<td>1.13</td>
</tr>
<tr>
<td>Average</td>
<td>1.28</td>
<td>1199.88</td>
<td>0.45</td>
<td>1163.18</td>
</tr>
</tbody>
</table>

The three algorithms found a feasible solution for every instance. On the one hand, regarding instances with 2 plants and disregarding the computational times, CPLEX clearly outperformed on average the proposed heuristics. On the other hand, LaPRe obtained the best results for the largest instances, being, on average, much better than CPLEX and Lag. Notice that for the instances with 6 plants, 25 and 50 items, CPLEX presented a very poor performance in comparison to its results for the other instances. In such cases, the gaps were higher than 3%, suggesting its limitations to solve large instances.

Table 2 displays, for each class (indicated in the first column), the average re-
sults achieved by CPLEX, LaPRe and Lag, considering the set of 480 instances. Each pair of columns referring to a solution method has two columns expressing, respectively, the average gap (AG (%)) and the average time to achieve the best solutions (AT (sec)).

Table 2: Results obtained by CPLEX, LaPRe and Lag for all instances proposed in [16]. For each class of instances, this table reports the average gaps obtained with the CPLEX lower bound (AG) and the average times (AT), in seconds, to find the best solutions.

<table>
<thead>
<tr>
<th>Class</th>
<th>CPLEX</th>
<th>LaPRe</th>
<th>Lag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AG (%)</td>
<td>AT (sec)</td>
<td>AG (%)</td>
</tr>
<tr>
<td>THH</td>
<td>0.90</td>
<td>1684.90</td>
<td>0.85</td>
</tr>
<tr>
<td>THL</td>
<td>7.88</td>
<td>1667.73</td>
<td>1.22</td>
</tr>
<tr>
<td>TLH</td>
<td>0.02</td>
<td>905.73</td>
<td>0.06</td>
</tr>
<tr>
<td>TLL</td>
<td>0.04</td>
<td>1194.03</td>
<td>0.09</td>
</tr>
<tr>
<td>NHH</td>
<td>0.63</td>
<td>1504.85</td>
<td>0.64</td>
</tr>
<tr>
<td>NHL</td>
<td>0.77</td>
<td>1592.98</td>
<td>0.70</td>
</tr>
<tr>
<td>NLH</td>
<td>0.01</td>
<td>434.43</td>
<td>0.02</td>
</tr>
<tr>
<td>NLL</td>
<td>0.01</td>
<td>614.40</td>
<td>0.03</td>
</tr>
<tr>
<td>Average</td>
<td>1.28</td>
<td>1199.88</td>
<td>0.45</td>
</tr>
</tbody>
</table>

By Table 2, one can observe that, except for NHH, LaPRe was considerably better on average than the other solution methods in the classes of instances with high setup cost. In particular, CPLEX did not perform well for the class THL, with an average gap of 7.88%. For all classes of instances with low setup costs both Lag and LaPRe achieved their best gaps, even though their average gaps were higher than CPLEX’s. For these classes, the highest gap of LaPRe was 0.09%, against 1.01% from Lag.

In a second part of this experiment, the average results of the three heuristics, Lag, LaPRe and GPheur, are compared. In Table 3, columns ‘Fea’ indicate the percentage of instances for which the corresponding algorithms found a feasible solution. Table 3 also shows the average gaps (AG (%)) and average times to achieve the best solutions (AT (sec)). For a fair comparison, the average results reported in Table 3 are solely of the instances for which GPheur had feasible solutions.

In Table 3, one can observe that the proposed heuristics were significantly
Table 3: Heuristic results obtained in Experiment I. For each solution method, this table presents the feasibility rate of the algorithm (Fea), the average gaps (AG) and the average times (AT), in seconds, to solve each class of instances.

<table>
<thead>
<tr>
<th>Class</th>
<th>Fea (%)</th>
<th>AG (%)</th>
<th>AT (sec)</th>
<th>Fea (%)</th>
<th>AG (%)</th>
<th>AT (sec)</th>
<th>Fea (%)</th>
<th>AG (%)</th>
<th>AT (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LaPRe</td>
<td></td>
<td></td>
<td>Lag</td>
<td></td>
<td></td>
<td>GPheur</td>
<td></td>
<td></td>
</tr>
<tr>
<td>THH</td>
<td>100.00</td>
<td>0.84</td>
<td>1387.55</td>
<td>100.00</td>
<td>5.00</td>
<td>1158.10</td>
<td>81.67</td>
<td>2.73</td>
<td>1442.80</td>
</tr>
<tr>
<td>THL</td>
<td>100.00</td>
<td>1.23</td>
<td>1365.15</td>
<td>100.00</td>
<td>5.77</td>
<td>971.35</td>
<td>80.00</td>
<td>3.51</td>
<td>1547.73</td>
</tr>
<tr>
<td>TLH</td>
<td>100.00</td>
<td>0.06</td>
<td>1077.49</td>
<td>100.00</td>
<td>0.75</td>
<td>1094.43</td>
<td>88.33</td>
<td>1.15</td>
<td>1397.68</td>
</tr>
<tr>
<td>TLL</td>
<td>100.00</td>
<td>0.08</td>
<td>1129.49</td>
<td>100.00</td>
<td>0.89</td>
<td>1070.67</td>
<td>85.00</td>
<td>1.28</td>
<td>1347.39</td>
</tr>
<tr>
<td>NHH</td>
<td>100.00</td>
<td>0.63</td>
<td>1183.65</td>
<td>100.00</td>
<td>2.96</td>
<td>1008.60</td>
<td>95.00</td>
<td>1.76</td>
<td>1578.89</td>
</tr>
<tr>
<td>NHL</td>
<td>100.00</td>
<td>0.70</td>
<td>1277.27</td>
<td>100.00</td>
<td>3.45</td>
<td>971.09</td>
<td>91.67</td>
<td>2.12</td>
<td>1412.38</td>
</tr>
<tr>
<td>NLH</td>
<td>100.00</td>
<td>0.02</td>
<td>789.92</td>
<td>100.00</td>
<td>0.30</td>
<td>951.90</td>
<td>100.00</td>
<td>0.46</td>
<td>1446.25</td>
</tr>
<tr>
<td>NLL</td>
<td>100.00</td>
<td>0.03</td>
<td>1023.97</td>
<td>100.00</td>
<td>0.36</td>
<td>1046.57</td>
<td>100.00</td>
<td>0.61</td>
<td>1380.68</td>
</tr>
<tr>
<td>Average</td>
<td>100.00</td>
<td>0.45</td>
<td>1154.31</td>
<td>100.00</td>
<td>2.43</td>
<td>1034.09</td>
<td>90.21</td>
<td>1.70</td>
<td>1444.22</td>
</tr>
</tbody>
</table>

Besides the lower gaps, the feasibility rate of GPheur indicates it does not find feasible solutions for at least 5% of the instances of the 5 classes. Additionally, LaPRe and Lag presented computational times inferior than GPheur’s.

Figure 2 displays the performance profiles, where ζ is composed by all four algorithms (Lag, LaPRe, GPheur and CPLEX), and Ψ by the benchmark instances. The left graphic 2(a) concerns the solution values, whereas the right graphic 2(b), the time to solve them. In particular, τ = 0 shows the percentage that an algorithm ζ outperformed the other algorithms.

(a) Performance profile according to the obtained solution values. (b) Performance profile according to the computational times.

Figure 2: Performance profiles for the small-sized instances.
The results indicate that, within the time limit of 30 minutes, CPLEX achieved the best results in more than 74% of the problems, in contrast to the 27% of LaPRe, 0% of Lag and 0.2% of GPheur. However, with a slack of less than 2% ($\tau$ approximately 0.02) on the quality of the best solutions, LaPRe curve crosses the curve of CPLEX. This means that 89% of the solution values of both algorithms were within the interval $[s_{\text{best}}, 1.02 \times s_{\text{best}}]$, where $s_{\text{best}}$ is the best solution of a given instance.

Concerning the time to find the best solution, CPLEX outperformed the other algorithms in 32.5% of the problems, LaPRe in 22.5%, Lag in 35.4% and GPheur in 9.0%. In both graphs, GPheur presented the worst performance.

5.2. Experiment II

This experiment evaluates the performance of the Lagrangian heuristics by contrasting, again, their results with those found by CPLEX and GPheur. The set of instances used for such are the large-sized instances discussed earlier in this section. In this experiment, the lower bounds are the best Lagrangian lower bounds, that were better than CPLEX’s, reported in https://sites.google.com/site/nascimentoemcv/downloads/mpc1sp. Initially, the problem instances were modeled according to the integer program (1)-(6). However, for these instances, CPLEX required a high memory storage and halt before returning any solution. Therefore, in this experiment, the CPLEX solver attempted to identify solutions of the problem instances modeled with the mathematical formulation described in [18]. The strongest bounds were found by the Lagrangian heuristics, that presented a feasible solution for all instances tested. Therefore, the lower bounds used to calculate the gaps were from the Lagrangian relaxation (guided by LaPRe).

First, only the results of CPLEX and the Lagrangian heuristics are presented, since both CPLEX and GPheur did not find feasible solutions for all instances. In line with this, Table 4 shows the average gaps and times of CPLEX, Lag and LaPRe. For computing these averages, only the solutions of the instances found by the three algorithms were considered.
Table 4: Results obtained by CPLEX, LaPRe and Lag for all instances proposed in [16]. For each class of instances, this table reports the feasibility rate of the algorithm (Fea), the average gaps obtained with the LaPRe lower bounds (AG) and the average times (AT), in seconds, to solve each class of instances.

<table>
<thead>
<tr>
<th>Class</th>
<th>CPLEX</th>
<th>LaPRe</th>
<th>Lag</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fea (%)</td>
<td>AG (%)</td>
<td>AT (sec)</td>
</tr>
<tr>
<td>THH</td>
<td>70.00</td>
<td>82.54</td>
<td>483.48</td>
</tr>
<tr>
<td>THL</td>
<td>67.00</td>
<td>82.84</td>
<td>552.50</td>
</tr>
<tr>
<td>TLH</td>
<td>80.00</td>
<td>29.33</td>
<td>1393.21</td>
</tr>
<tr>
<td>TLL</td>
<td>50.00</td>
<td>43.60</td>
<td>1113.53</td>
</tr>
<tr>
<td>NHH</td>
<td>83.00</td>
<td>83.74</td>
<td>841.92</td>
</tr>
<tr>
<td>NHL</td>
<td>83.00</td>
<td>84.06</td>
<td>992.80</td>
</tr>
<tr>
<td>NLH</td>
<td>63.00</td>
<td>29.91</td>
<td>1132.63</td>
</tr>
<tr>
<td>NLL</td>
<td>60.00</td>
<td>41.31</td>
<td>1462.50</td>
</tr>
<tr>
<td>Average</td>
<td>69.58</td>
<td>59.66</td>
<td>996.57</td>
</tr>
</tbody>
</table>

By Table 4, one may observe an outstanding performance of both Lagrangian heuristics, being significantly better than CPLEX for every class of instances. Again, the setup cost had a major influence in the solution quality. For the classes with high setup cost, LaPRe encountered solutions with average gaps lower than or equal to 1.61%. The average gaps of the classes with low setup cost were not superior to 0.68%. On the other hand, CPLEX had a poor performance with regard to both feasibility rate and to the quality of the solutions found. Even though CPLEX obtained more feasible solutions to those instances with high setup costs, it presented very high average gaps, between 82% to 84%. For the classes with low setup costs, CPLEX had the worst feasibility rate. Concerning the average gaps, they were within the range from 29.33% to 43.6%, significantly superior to the highest gap of the Lagrangian heuristics, 0.98%.

Table 5 reports the average gaps and times to find the best solutions by the three heuristic methods. This table solely reports the averages taking into consideration the instances to which solutions were found by the three algorithms.

GPheur presented 100% feasibility rate only for the instances of the classes NLH and NLL. Among the heuristics, LaPRe was the one that achieved the best solutions. It considerably outperformed GPheur and had better average
Table 5: Results obtained in Experiment II. For each solution method, this table presents the feasibility rate of the algorithm (Fea), the average gaps (AG) and the average times (AT), in seconds, to solve each class of instances.

<table>
<thead>
<tr>
<th>Class</th>
<th>LaPRe Fea (%)</th>
<th>Lag Fea (%)</th>
<th>GPheur Fea (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AG (%)</td>
<td>AT (sec)</td>
<td>AG (%)</td>
</tr>
<tr>
<td></td>
<td>AT (sec)</td>
<td></td>
<td>AT (sec)</td>
</tr>
<tr>
<td>THH</td>
<td>100.00 0.99</td>
<td>100.00 1.90</td>
<td>100.00 5.15</td>
</tr>
<tr>
<td>TLL</td>
<td>100.00 1.66</td>
<td>100.00 2.22</td>
<td>100.00 5.62</td>
</tr>
<tr>
<td>TLH</td>
<td>100.00 0.22</td>
<td>100.00 0.56</td>
<td>100.00 6.00</td>
</tr>
<tr>
<td>TLL</td>
<td>100.00 0.45</td>
<td>100.00 0.89</td>
<td>53.33 4.81</td>
</tr>
<tr>
<td>NHH</td>
<td>100.00 0.42</td>
<td>100.00 0.97</td>
<td>100.00 4.39</td>
</tr>
<tr>
<td>NFL</td>
<td>100.00 0.71</td>
<td>100.00 1.35</td>
<td>100.00 4.74</td>
</tr>
<tr>
<td>NLL</td>
<td>100.00 0.40</td>
<td>100.00 0.74</td>
<td>100.00 4.55</td>
</tr>
<tr>
<td>NLL</td>
<td>100.00 0.14</td>
<td>100.00 0.27</td>
<td>100.00 3.53</td>
</tr>
<tr>
<td>Average</td>
<td>100.00 0.62</td>
<td>100.00 1.11</td>
<td>84.16 4.68</td>
</tr>
</tbody>
</table>

gaps in all classes of instances. All these results attest a clear superiority of the proposed strategies for large instances.

Finally, Table 6 shows the average results achieved by the Lagrangian heuristics for large instances.

Table 6: Average results obtained by the Lagrangian heuristics for the large-sized instances.

<table>
<thead>
<tr>
<th>Class</th>
<th>LaPRe AG (%)</th>
<th>Lag AG (%)</th>
<th>LaPRe AT (sec)</th>
<th>Lag AT (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>THH</td>
<td>0.99 1105.83</td>
<td>1.90 1177.70</td>
<td>100.00 1.90</td>
<td>1177.70</td>
</tr>
<tr>
<td>THL</td>
<td>1.66 1091.50</td>
<td>2.22 1176.37</td>
<td>100.00 2.22</td>
<td>1176.37</td>
</tr>
<tr>
<td>TLH</td>
<td>0.29 1245.60</td>
<td>0.62 1259.70</td>
<td>100.00 0.56</td>
<td>1214.22</td>
</tr>
<tr>
<td>TLL</td>
<td>0.57 914.30</td>
<td>0.91 1412.20</td>
<td>100.00 0.89</td>
<td>1346.69</td>
</tr>
<tr>
<td>NHH</td>
<td>0.42 1279.67</td>
<td>0.97 1046.17</td>
<td>100.00 0.97</td>
<td>1046.17</td>
</tr>
<tr>
<td>NHL</td>
<td>0.71 1171.77</td>
<td>1.35 1060.63</td>
<td>100.00 1.35</td>
<td>1060.63</td>
</tr>
<tr>
<td>NLH</td>
<td>0.63 1207.50</td>
<td>0.81 1331.03</td>
<td>100.00 0.74</td>
<td>1320.17</td>
</tr>
<tr>
<td>NLL</td>
<td>0.14 1011.60</td>
<td>0.27 1289.73</td>
<td>100.00 0.27</td>
<td>1289.73</td>
</tr>
<tr>
<td>Average</td>
<td>0.68 1150.97</td>
<td>1.13 1219.19</td>
<td>100.00 1.11</td>
<td>1203.96</td>
</tr>
</tbody>
</table>

To a further assessment of the results, Figure 3 shows the performance profiles as in Experiment I. In this case, ζ is composed by the benchmark large-instances, whereas Ψ contains CPLEX, Lag, LaPRe and GPheur. Figures 3(a) and 3(b) display the profiles with regard to, respectively, the solution values and the time required to find their best solutions.
6. Final Remarks

This paper approaches the multi-plant capacitated lot sizing problem (MPCLSP). Its relevance is well-grounded since it can be found in different industrial sectors, to determine the best production planning. There is a small body of literature that focuses on the MPCLSP. In particular, a GRASP with path-relinking (GPheur) [16], is the most recent heuristic for this problem and outperformed previous solution methods. Nevertheless, GPheur did not find feasible solutions for a number of benchmark instances. Therefore, the need...
for efficient methods to solve the MPCLSP is the primary reason behind the proposal of this paper, novel Lagrangian heuristics.

In addition to determining good quality upper bounds, this paper puts forward the Lagrangian strategies to achieve good lower bounds for the tested instances. In a first experiment, using the benchmark instances introduced in [16], the Lagrangian heuristics significantly outperformed the average results of CPLEX, within the time limit of 1800 seconds, for some specific classes of instances: those with high setup costs. In comparison to GPheur, the Lagrangian heuristics clearly showed a better performance, considering solution values and times. It is worth mentioning that the Lagrangian heuristics, unlike GPheur, found a feasible solution for every instance. In a second experiment, with large instances, the novel Lagrangian heuristics outperformed both CPLEX lower and upper bounds within the time limit of 1800 seconds. Again, the Lagrangian heuristics were consistently better than GPheur.

One may conclude that for large-scale instances, the proposed heuristic appears as an outstanding alternative for providing reliable solutions attested by the lower bounds produced by the Lagrangian relaxation. Even for the smallest instances LaPRE, the Lagrangian heuristic with path-relinking, was very competitive. These results bear out the major role of matheuristics in solving hard optimization problems.

7. Acknowledgments

The authors are grateful for the financial support of the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) and the Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP). In this research, the authors used computational resources of the Center of Applied Mathematics to Industry (CeMEAI) supported by FAPESP. Furthermore, the authors are indebted to anonymous reviewers whose constructive criticism resulted in a better presentation of the material.
References


Appendix A. Fine-tuning parameters

This appendix summarizes the results of simulations performed for justifying the parameter values adopted in the subgradient method of the proposed Lagrangian heuristics. Accordingly, 9 instances of each class of the benchmark instances were considered and the impact on the upper bounds’ quality was observed under different values for parameters $\sigma$ and $\eta$. In the literature, in particular, for Lagrangian heuristics, for the capacitated lot sizing problems, there are studies in which such values range from 0.75 to 1.75 for $\sigma$. For $\eta$, values ranging from 1.5 to 3.0. Therefore, the results achieved by LaPRe for the pairwise combination of $\sigma = \{0.5; 0.6; \ldots; 1.9\}$ and $\eta = \{1.5; 2.0; 2.5; 3.0\}$ were taken into account. The normalized average results obtained by LaPRe with these parameter values are displayed in Figure A.4.

![Figure A.4: Figure displaying the relation between the average gaps, $\sigma$ and $\eta$.](image)

As it may be observed, the algorithm obtained its best results when $\sigma$ was
within the interval [1.6;1.8] and \( \eta = 3.0 \). Then, the values chosen for \( \sigma \) and \( \eta \) were, respectively, 1.6 and 3.0. Additionally, as suggested in [13], the initial value for the parameter \( q_t \) was twice the dimension of the problem, \( n \times m \times a \). Then, at each \( q_t \) iterations, \( q_t \) diminished by a factor \( \eta \). If \( q_t \) achieves value 10, it will keep this value for the remaining iterations. These values and strategies are based on the Lagrangian heuristic of a very similar problem [24] that relied on studies performed in [13].

The initial values of \( \lambda_{jt}^0 \) were equal to 0, \( \forall j,t \). The initial value for \( \alpha^v \) is Equation (21), as showed in the paper. Besides that, the strategy considers a restarting mechanism to promote a diversification of the procedure. Henceforth, at each 150 iterations that a solution remains unchanged, the respective cyclic values for \( \alpha^v \) are considered: \( 10^{-3} \), \( 10^{-4} \), \( 10^{-5} \) and Equation (21). The large number used to estimate \( Z_{sup}^v \) in the case no feasible solution was found up to iteration \( v \) was \( 10^8 \).

In Feasibility Phase I, the parameter \( \rho \), responsible for penalizing the overtime capacity in an infeasible solution, was fitted to 50 as proposed in [16]. To reach this value, Nascimento et al. [16] took as reference the strategy used in [11] that suggested 50 as the initial value for \( \rho \). The dynamicity of \( \rho \) in the self adjusting penalty proposed in [11] was not required in [16]. Then, as in [16], the strategy assigned a static value for \( \rho \) that remained 50 in all iterations.

The stop criterion referred in Algorithm 1 regards the same number of iterations suggested in [16]: 1000. In Feasibility Phase II, the maximum number of runs is 10, the same stop criterion proposed in [24].