A Lagrangean Relaxation Algorithm for the Simple Plant Location Problem with Preferences

Xavier Cabezas  
The University of Edinburgh  
J.X.Cabezas@sms.ed.ac.uk  
Escuela Superior Politécnica del Litoral  
joxacabe@espol.edu.ec

Sergio García  
The University of Edinburgh  
Sergio.Garcia-Quiles@ed.ac.uk

May 2018

Abstract

The Simple Plant Location Problem with Order (SPLPO) is a variant of the Simple Plant Location Problem (SPLP), where the customers have preferences over the facilities which will serve them. In particular, customers define their preferences by ranking each of the potential facilities. Even though the SPLP has been widely studied in the literature, the SPLPO has been studied much less and the size of the instances that can be solved is very limited. In this paper, we propose a heuristic that uses a Lagrangean relaxation output as a starting point of a semi-Lagrangean relaxation algorithm to find good feasible solutions (often the optimal solution). We carry out a computational study to illustrate the good performance of our method.

Keywords: Simple Plant Location Problem, Lagrangean relaxation, semi-Lagrangean relaxation, Preferences.

1 Introduction

The Simple Plant Location Problem with Order (SPLPO) is a variant of the Simple Plant Location Problem (SPLP), where the customers have preferences over the facilities which will serve them. In particular, customers define their preferences by ranking each of the potential facilities.

Let \( I = \{1, \ldots, m\} \) be a set of customers and \( J = \{1, \ldots, n\} \) a set of possible sites for opening facilities. Unit costs \( c_{ij} \geq 0 \) for supplying the demand of customer \( i \) from facility \( j \) and costs \( f_j \geq 0 \) for opening a facility at \( j \) are also considered. It is said that \( k \) is \( i \)-worse than \( j \) if customer \( i \) prefers facility \( j \) to \( k \) and it is written as \( k <_i j \). We define \( W_{ij} = \{k \in J \mid k <_i j\} \) as the set of facilities \( k \) strictly \( i \)-worse than \( j \), its complement as \( \overline{W}_{ij} \) and \( W_{ij} \cup \{j\} \) as \( W_{ij}' \). Let \( x_{ij} \) be a decision variable that represents the fraction of the demand required by customer \( i \) and supplied by facility \( j \). Since no capacities are considered, this demand can always be covered completely by one single facility. Therefore, we can guarantee that there is an optimal solution where variables \( x_{ij} \) are in \( \{0, 1\} \). Let \( y_j \) be a binary variable such that \( y_j = 1 \) if a facility is open at location \( j \) and \( y_j = 0 \) otherwise.

The SPLPO formulation (Cánovas et al., 2006) is as follows:
\[
\begin{align*}
\text{Min} & \quad \sum_{i \in I} \sum_{j \in J} c_{ij} x_{ij} + \sum_{j \in J} f_{j} y_{j}, \\
\text{subject to} & \quad \sum_{j \in J} x_{ij} = 1, \quad \forall i \in I, \quad (1) \\
& \quad x_{ij} \leq y_{j}, \quad \forall i \in I, \forall j \in J, \quad (2) \\
& \quad \sum_{k \in W_{ij}} x_{ik} \geq y_{j}, \quad \forall i \in I, \forall j \in J, \quad (3) \\
& \quad x_{ij} \geq 0, \quad \forall i \in I, \forall j \in J, \quad (4) \\
& \quad y_{j} \in \{0, 1\}, \quad \forall j \in J. \quad (5)
\end{align*}
\]

Equalities (1) ensure that every customer \( i \) will be supplied by exactly one facility \( j \), they are called assignment constraints. Constraints (2) ensure that if a customer \( i \) is supplied by a facility \( j \) then \( j \) must be opened, they are usually called variable upper bounds (VUBs). Inequalities (3) model the customers’ preference orderings.

There are in the literature many studies on how to solve the SPLP using different methods and lagrangean relaxation is one of the most successful. For example, in Cornuéjols et al. (1977), the authors presented an application of Lagrangean relaxation to solve the SPLP in the context of location of bank accounts. They also proposed a heuristic algorithm and studied the Lagrangean dual to obtain lower and upper bounds. They also provided a bound for the relative error of these methods. Beltrán et al. (2006) suggested, defined and applied the technique of semi-Lagrangean relaxation to the \( p \)-median problem. Some years later the study was extended to the SPLP, obtaining very good results. The method basically takes advantage of the linear formulation of the problems. First, it splits equality constraints \( Ax = b \) into \( Ax \leq b \) and \( Ax \geq b \), and then relaxes the second one. The new model has the same objective value as the original problem (it closes the duality gap), but with the cost of making the new problem more difficult to solve. However, it has some properties that can be exploited. A summary of this method will be reviewed later. Another reduced form of this method was proposed by Monabbati (2014), which called it a surrogate semi-Lagrangean relaxation. A new algorithm for the dual problem using Lagrangean heuristics for both the original and surrogate version of the semi-Lagrangean relaxation can be found in Jörnsten (2016).

The SPLPO has been studied much less and the main results are on finding new valid inequalities to strengthen the original formulation. Cánovas et al. (2006) provided a new family of valid constraints which were used in combination with a preprocessing analysis. Another, more general, family of valid inequalities can be found in Vasiliev et al. (2013) with a polyhedral study. A branch and cut method was proposed by Vasiliev and Klimentova (2010).

Since these papers use exact methods that struggle to solve large instances, the aim of this paper is to develop a procedure that allows to solve the SPLPO efficiently in a heuristic way by using Lagrangean and semi-Lagrangean relaxation techniques. We propose a variable fixing heuristic that uses a Lagrangean relaxation output as the starting point of a semi-Lagrangean relaxation to find good feasible solutions (often the optimal solution).

The rest of the paper is organized as follows: Section 2 provides a review of Lagrangean and semi-Lagrangean relaxation. SPLPO formulations for these two techniques are showed in Section 3 and 4, along with the methods that will be used later to solve their respective dual problems. The complete method that we propose is presented in Section 5, where all the algorithms proposed in previous sections will be combined to build a heuristic procedure. In Section 6 we run a computational study that shows the good performance of our method. Finally, some conclusions are given in Section 7.
2 Lagrangean and Semi-Lagrangean Relaxation Background

Let $P$ be a problem of the form:

$$
\min_x \{ f(x) = cx \mid Ax \leq b, Cx \leq d, x \in X \},
$$

where:

- The set $X$ may contain integrality constraints.
- The family of constraints $Ax \leq b$ will be assumed complicated. (i.e., the problem $P$ without them is easier to solve).

The family $Ax \leq b$ can be placed in the objective function with vector of coefficients (Lagrangean multipliers) $\lambda$ which work as penalties when they are violated. It yields the Lagrangean relaxation problem $LR(\lambda)$ related to $P$ with multipliers $\lambda$:

$$
LR(\lambda) = \min_x \{ f(x) + \lambda(Ax - b) \mid Cx \leq d, x \in X, \lambda \geq 0 \},
$$

with its associated Lagrangean dual problem:

$$
LD(\lambda) = \max_{\lambda \geq 0} \min_x \{ f(x) + \lambda(Ax - b) \mid Cx \leq d, x \in X, \lambda \geq 0 \}
= \max_{\lambda \geq 0} LR(\lambda).
$$

Let $v(p)$ be the optimal objective function value for a particular problem $p$. For all $x \in X$ feasible for $P$ and any $\lambda \geq 0$, $f(x) + \lambda(Ax - b) \leq f(x)$. Therefore, $v(LD(\lambda)) \leq v(P)$ and $v(LR(\lambda)) \leq v(P)$. Furthermore, if we supposed that $\{x \in X \mid Cx \leq X\}$ is a bounded polyhedron, there is a finite family of extreme points $x$ in its the convex hull and therefore the objective function of $LR(\lambda)$ is a piecewise linear concave function on $\lambda$. Thus, there are some points where the function is not differentiable. Additionally, it can be proved that if $x^*$ be an optimal solution for $LR(\lambda^*)$, $Ax^* - b$ is a subgradient of $LR(\lambda)$ at point $\lambda^*$.

Also, if $LP(P)$ is problem $P$ without integrality constraints (that is, its linear relaxation), then $v(LP(P)) \leq v(LD(\lambda)) \leq v(P)$. And, $v(LP(P)) = v(LD(\lambda))$ whenever $v(LP(LD(\lambda))) = v(LD(\lambda))$ (which is called integrality property).

Consider now the problem $P'$:

$$
\min_x \{ f(x) = cx \mid Ax = b, x \in X \},
$$

where:

- The set $X$ contains integrality constraints.
- $A$, $b$ and $c$ are non-negative.

The semi-lagrangean relaxation problem $SLR(\lambda)$ related to $P'$ is:

$$
SLR(\lambda) = \min_x \{ f(x) + \lambda(b - Ax) \mid Ax \leq b, x \in X, \lambda \geq 0 \}.
$$

After having split $Ax = b$, inequality $Ax \geq b$ has been relaxed with a vector multipliers $\lambda$ whereas inequality $Ax \leq b$ has been kept. Its semi-Lagrangean dual problem is then:

$$
SLD(\lambda) = \max_{\lambda \geq 0} SLR(\lambda).
$$

It is clear that $v(LR(\lambda)) \leq v(SLR(\lambda))$ because $SLR(\lambda)$ has more constraints. Thus, $v(LP(P')) \leq v(SLD(\lambda)) \leq v(SLR(\lambda)) \leq v(P')$. Beltrán et al. (2006) proved that the semi-Lagrangean dual problem has the same optimal value than $P'$, i.e., $v(P') = v(SLD(\lambda))$. They also proved that the objective function of $SLR(\lambda)$ is concave, non-decreasing on its domain and $b - Ax$ is a subgradient at point $\lambda$. Moreover, there is an interval $[\lambda^*, +\infty)$ where with any of its elements we met the same optimal solution of $SLR(\lambda)$. For details, see Geoffrion (1974), Fisher (2004), Beltrán et al. (2006, 2012).
3 A Lagrangean Relaxation for SPLPO

In this section we consider a Lagrangean Relaxation for the SPLPO. Our first result shows that the proposed model is easy to solve.

If we relax constraints (1) and (3), then they are moved to the objective function with penalty coefficients (multipliers) when they are violated, thus obtaining the following Lagrangean relaxation problem $LR(\mu, \lambda)$:

$$
LR(\mu, \lambda) = \min_{(x,y)} \left( \sum_i \sum_j c_{ij} x_{ij} + \sum_j f_j y_j + \sum_i \mu_i \left( 1 - \sum_j x_{ij} \right) + \sum_i \sum_j \lambda_{ij} \left[ y_j - \sum_{k \in W_{ij}} x_{ik} \right] \right),
$$

subject to: (2), (4) and (5).

The multiplier vectors $\mu$ and $\lambda$ in $LR(\mu, \lambda)$ are unrestricted in sign and nonnegative, respectively.

Let $F(P)$ be the set of feasible solutions of problem $P$. Then for all $(x, y) \in F(SPLPO)$ the objective function of $LR(\mu, \lambda)$ evaluated in $(x, y)$ is always less than or equal to the objective function of $P$ evaluated in $(x, y)$. Therefore $v(LR(\mu, \lambda)) \leq v(SPLPO)$.

In order to obtain the best lower bound for SPLPO, we need to solve the following Lagrangean dual problem:

$$
LD_{\mu \lambda} = \max_{\mu \in \mathbb{R}, \lambda \geq 0} LR(\mu, \lambda).
$$

Suppose that each customer $i$ ranks the different potential facilities $j$ with a number $p_{ij} \in \{1, \ldots, n\}$ with 1 and $n$ the most and the least preferred, respectively. Since each multiplier $\lambda_{ij}$ in a term of $\sum_i \sum_j \lambda_{ij} \sum_{k \in W_{ij}} x_{ik}$ in $LR(\mu, \lambda)$ will be multiplied by a sum of $p_{ij}$ variables $x_{ik}$ with $k \geq i$ $j$, then each $x_{ij}$ will be multiplied by a sum of $(n - p_{ij} + 1)$ values $\lambda_{ik}$ with $k \leq i$ $j$. Therefore:

$$
\sum_i \sum_j \lambda_{ij} \sum_{k \in W_{ij}} x_{ik} = \sum_i \sum_j \left( \sum_{k \in W'_{ij}} \lambda_{ik} \right) x_{ij},
$$

and $LR(\mu, \lambda)$ can be rewritten as:

$$
LR(\mu, \lambda) = \min_{(x,y)} \sum_i \sum_j \left( c_{ij} - \mu_i - \sum_{k \in W'_{ij}} \lambda_{ik} \right) x_{ij} + \sum_j \left( f_j + \sum_i \lambda_{ij} \right) y_j + \sum_i \mu_i,
$$

subject to: (2), (4) and (5).

As in Cornuéjols et al. (1977), this Lagrangean problem is easy to solve analytically for fixed vectors $\mu$ and $\lambda$. If we define $\Lambda_{ij}$ as:

$$
\Lambda_{ij} = \sum_{k \in W'_{ij}} \lambda_{ik},
$$

then we have the following result:

**Theorem 1.** An optimal solution for $LR(\mu, \lambda)$ can be obtained as follows:

$$
y_j = \begin{cases} 
1, & \text{if } \sum_i \min(0, c_{ij} - \mu_i - \Lambda_{ij}) + (f_j + \sum_i \lambda_{ij}) < 0, \\
0, & \text{otherwise}.
\end{cases}
$$
and,

\[ x_{ij} = \begin{cases} 
1, & \text{if } y_j = 1 \text{ and } (c_{ij} - \mu_i - \Lambda_{ij}) < 0, \\
0, & \text{otherwise}. 
\end{cases} \]

Proof. We have that:

\[
LR(\mu, \lambda) = \min_{(x, y)} \sum_i \sum_j \left( c_{ij} - \mu_i - \Lambda_{ij} \right) x_{ij} + \sum_j \left( f_j + \sum_i \lambda_{ij} \right) y_j + \sum_i \mu_i,
\]

subject to: (2), (4) and (5).

As a consequence of constraints (2) and for fixed vectors \( \mu \) and \( \lambda \), the optimal values for \( x_{ij} \) will be \( x_{ij} = 1 \) if \( y_j = 1 \) and \( c_{ij} - \mu_i - \Lambda_{ij} < 0 \), otherwise \( x_{ij} = 0 \). Then, if we define 

\[ \rho_j(\mu, \lambda) = \sum_i \min(0, c_{ij} - \mu_i - \Lambda_{ij}) + (f_j + \sum_i \lambda_{ij}), \]

the optimal vector \( y \) can be obtained by solving the following minimization problem:

\[
\min \sum_j \rho_j(\mu, \lambda) y_j,
\]

subject to: \( y_j \in \{0, 1\} \).

The solution to this problem is straightforward.

An issue with this model is that \( LR(\mu, \lambda) \) has the integrality property, that is, its optimal value is equal to the standard linear relaxation \( LP(\text{SPLPO}) \). Furthermore, the values obtained of the function \( LR(\mu, \lambda) \) during the search of the solution for \( LD_{\mu\lambda} \) are infeasible to the original problem \( \text{SPLPO} \). So, as an alternative, we propose to use the solution of this problem as a starting point for another procedure that allows us to find feasible solutions to \( \text{SPLPO} \).

### 3.1 Subgradient Method for the Lagrangean Dual \( LD_{\mu\lambda} \)

The subgradient method was originally proposed by Held and Karp (1971) and validated by Held et al. (1974). Given multipliers \( \lambda \) and \( \mu \), this method tries to optimize \( LD \) by taking steps along a subgradient of \( LR(\mu, \lambda) \). A sketch of the whole procedure is given in Algorithm 1.

As seen in Steps 2 and 5, at each iteration, we need to solve an instance of \( LR(\mu, \lambda) \). Each of them is easy to solve, as shown in Theorem 1. However, it is important to note that the whole procedure could still be slow, that is, slower than solving \( LP(\text{SPLPO}) \) with a commercial linear programming solver.

It is easy to see that:

\[
s = \begin{bmatrix}
1 - \sum_{j \in J} x_{ij}; \forall i \\
y_j \sum_{k \in W_{ij}} x_{ik}; \forall i, j
\end{bmatrix} = \begin{bmatrix}
s_p \\
s_{s}\lambda
\end{bmatrix} \in \mathbb{R}^{m(n+1)},
\]

is a subgradient for \( LR(\mu, \lambda) \). If this vector is 0, then the procedure ends.

In our computational experiments, that will be showed later, an upper bound for \( LR_{AIM} \) in Step 5 is found using a simple heuristic. See Algorithm 2. First, it opens a facility that supplies all customers with the lowest operating cost and it is removed from the set \( J' = J \). Then, for each unopened one, each customer compares and chooses the most preferred facility between it and its previously assigned supplier. The new cost is saved. The new open facility is the one with the lowest operating cost. It is removed from \( J' \). This is repeated until \( J' = \{ \} \).
**Algorithm 1** Subgradient method (SG) for SPLPO.

Let $LD = \max(\mu, \lambda)LR(\mu, \lambda)$ with $\mu \in \mathbb{R}$ and $\lambda \geq 0$.

Step 1. (Initialization). Let $LR_{AIM}$ by a heuristic method. Set $\beta = 2$. Let $k$ be an integer number. Let $q$ be a number in $[0, 1]$. Let $[\mu^0, \lambda^0]$ be a starting point.

Step 2. (Obtaining values $x_{ij}, \forall i, j$ and $y_{ij}, \forall j$). Find $LR_{best}^0 = LR(\mu^0, \lambda^0)$. Set $iter = 0$.

Step 3. (Finding a subgradient). Find a subgradient $s_{iter}$ for $LR(\mu_{iter}, \lambda_{iter})$.

Step 4. (Stop criterion). If $s_{iter} = 0$, STOP and $[\mu_{iter}, \lambda_{iter}]$ is optimal. Otherwise, go to Step 5.

Step 5. (Step size). Let $\alpha_{iter} = \beta \frac{LR_{AIM} - LR_{iter+1}^0}{\|\sum_{ij} \alpha_{iter} s_{iter}\|_2}$.

If $LR(\mu_{iter}, \lambda_{iter})$ does not improve for $k$ consecutive iterations, set $\beta = \beta \times q$.

Step 6. (Updating multipliers) Set $[\mu_{iter+1}, \lambda_{iter+1}] = (\mu_{iter} + \alpha_{iter} s_{iter}, \max(0, \lambda_{iter} + \alpha_{iter} s_{iter}))$.

Step 7. (Updating incumbent). Let $LR_{best}^{iter+1} = \max\{LR_{best}^iter, LR(\mu_{iter+1}, \lambda_{iter+1})\}$. Update $iter = iter + 1$.

Step 8. (Stop criterion). If $iter = MAXiter$, STOP. Otherwise go to Step 3.

**Algorithm 2** Heuristic to find an upper bound for SPLPO (Hc).

Let $G(I \times J, E = \{\})$ be a bipartite graph.

Step 1. Find $j_0 \in J$ such that $\sum_{i \in I} c_{ij_0} = \min\{\sum_{i \in I} c_{i1}, \ldots, \sum_{i \in I} c_{in}\}$.

Set $J' = J \setminus \{j_0\}$, $E = \{(i, j_0)\}$ and $TC_{prev} = \sum_{i \in I} c_{ij_0}$.

Step 2. For all $j \in J'$, do:

Set $E_j = \{\}$.

For all $i \in I$, do:

Find $k_{pref}$ such that $p_{k_{pref}} = \min\{\{p_{ij} \cup \{p_{ik} \mid (i, k) \in E\}\}$.

Set $E_j = E_j \cup \{(k_{pref})\}$.

Compute $TC_j = \sum_{(i, j) \in E_j} c_{ij}$.

Step 3. Find $j_0$ such that $TC_{j_0} = \min\{TC_j \mid j = 1, \ldots, n\}$. Set $J' = J' \setminus \{j_0\}$ and $E = E_{j_0}$.

Step 4. If $J' = \{\}$, STOP. Otherwise go to Step 2.

We also tried another heuristic. It is basically the same that for Hc with a different Step 4 which allows the algorithm to stop earlier. We name it Hs (see Algorithm 3) and its results will be reported later.

**Algorithm 3** Hs.

Step 1..3. Same as Hc.

Step 4. If $CT_{j_0} \geq CT_{prev}$ then STOP. Otherwise, set $CT_{prev} = CT_{j_0}$ and go to Step 2.

It is possible, due to an overestimation of $LR_{AIM}$, that the function $LR(\mu, \lambda)$ does not improve for many iterations. This can be overcome by setting the parameter $\beta$ to a fixed value (for example, 2) and reducing it slowly.

In Step 6, we need vector $\lambda$ to be nonnegative, therefore in the nonnegative orthant, i.e., $[\lambda^+] = \max\{0, \lambda\}$, for all of its components. $\mu$ must remain unchanged as it is an unrestricted vector.

Further details can be found in Guignard (2003, Conforti et al., 2014) and Poljak (1967) for the step size in Step 5 in Algorithm 1.
4 A Semi-Lagrangean Relaxation for SPLPO

Now we use the technique of the Semi-Lagrangean relaxation, as this leads to close the duality gap. The equality constraints (1) have been split into two inequalities \( \sum_{j \in J} x_{ij} \leq 1 \) and \( \sum_{j \in J} x_{ij} \geq 1 \) to obtain the following model:

\[
SLR(\gamma) = \min_{(x,y)} \sum_i \sum_j c_{ij} x_{ij} + \sum_j f_j y_j + \sum_i \gamma_i \left(1 - \sum_j x_{ij}\right),
\]

\[
= \min_{(x,y)} \sum_i \sum_j (c_{ij} - \gamma_i) x_{ij} + \sum_j f_j y_j + \sum_i \gamma_i,
\]

\[
= \min_{(x,y)} \sum_j \left(\sum_i (c_{ij} - \gamma_i) x_{ij} + f_j y_j\right) + \sum_i \gamma_i,
\]

subject to: (2), (3), (4), (5), and \( \sum_{j \in J} x_{ij} \leq 1 \). Every component of the multipliers vector \( \gamma \) is nonnegative.

As mentioned in Section 2, the objective function is concave and non-decreasing in its domain. Therefore, its semi-Lagrangean dual problem:

\[
SLD_\gamma = \max_{\gamma \geq 0} SLR(\gamma),
\]

can be solved using an ascent method. Also, as pointed out, there is a set \( Q = [\gamma^*, +\infty) \) such that, for any \( q \in Q \) the optimum of \( SLR(\gamma) \) is met.

For the SPLP (no preferences), Beltrán et al. (2012) proved that there is a closed interval where the search of the multipliers could be done. Following the same idea, we provide the next two results for the SPLPO:

**Theorem 2.** Let \( cp_i = \max_j \{c_{ij} + f_j\} \) and let \( cp = (cp_1, \ldots, cp_m) \) be the maximum of the costs for each customer \( i \) associated to each facility \( j \) and the vector of these costs, respectively. If \( \gamma \geq cp \), then \( \gamma \in Q \).

**Proof.** As we know, the semi-Lagrangean relaxation closes the duality gap if for all \( i \), \( \sum_{j \in J} x_{ij} = 1 \). Assume that \( SLR(\gamma) < \infty \), otherwise the proposition is trivially true. By hypothesis, \( cp_i - \gamma_i \leq 0 \) for all \( i \in I \). If we choose \( j' \) such that \( cp_i = c_{ij'} + f_{j'} \), it turns out in \( (c_{ij'} - \gamma_i) + f_{j'} \leq 0 \), and this inequality is true for any \( j \) since \( j' \) gives the maximum among all \( c_{ij} + f_j \). Therefore, the event \( \sum_{j \in J} x_{ij} = 0 \) can not happen at an optimal solution, because it is always possible set \( x_{ij} = 1 \) and \( y_j = 1 \) for all \( i \) and \( j \) meeting all the constraints. \( \Box \)

**Theorem 3.** For each \( i \in I \), let \( c_i^1 \leq \ldots \leq c_i^n \) be the sorted costs \( c_{ij} \). If \( \gamma < c_i^1 \) then \( \gamma \notin Q \).

**Proof.** By hypothesis \( c_i^1 - \gamma_i > 0 \), then \( c_i^j - \gamma_i > 0 \) for all \( j \in J \). In that case, \( x_{ij}^* = 0 \) for all \( j \in J \) in any optimal solution \( x_\gamma^* \). Therefore, \( \sum_{j \in J} x_{ij} = 1 \) can not happen at an optimal solution and \( \gamma \notin Q \). \( \Box \)

The result of Theorem 2 is weaker than the obtained by Beltrán et al. (2012) for SPLP in the following sense. They choose \( cp_i \) equal to \( \min_{j} \{c_{ij} + f_j\} \), but with this, there is no guarantee that the preference constraints hold. We can set \( x_{ij}^* = 1 \) and \( y_j = 1 \) for the particular \( j' \) such that \( \min_{j} \{c_{ij} + f_j\} = c_{ij'} + f_{j'} \) but not necessarily for all \( j \), as in this way, are not available all the possible combinations of \( x \)'s and \( y \)'s that consider all customers preferences. Therefore, we are taking the risk of obtaining a non optimal solution.

The interval of search for \( \gamma \) components is then:

\[
B = \{\gamma \mid c^1 < \gamma \leq cp\}.
\]
Using the sorted costs $c_i^1 \leq \ldots \leq c_i^n$, each component $\gamma_i$ of $\gamma$ can be either in an interval of the form $I_j = (c_i^j, c_i^{j+1})$ where $j \in \{1, \ldots, m-1\}$ or out of it. For the first case, there are infinite values of $\gamma_i$ that can belong to a single interval $(c_i^j, c_i^{j+1})$, but each of them has the same effect on the optimal value of $SLR(\gamma)$. This is because going from an interval $I_j$ to the next $I_{j+1}$ could imply a change in the choice of the arc $(i, j)$ to the arc $(i, j+1)$ in the bipartite graph $G_\gamma = (I \times J, E = \{(i, j) \mid x_{ij} = 1 \mbox{ in the solution of } SLR(\gamma)\})$, which means a change in the solution. Hence, we just need to single $\gamma_i$ representative of the intervals. As $\gamma$ goes to infinity all combined costs $(c_{ij} - \gamma_i) + f_j$ become negative, and hence the semi-Lagrangian relaxation problem can be as difficult to solve as the original SPLPO problem. Then, it is always convenient to choose a $\gamma_i \in I_i$ as smaller as possible, that is, at an epsilon $\epsilon$ distance from the lower bound of an interval. Also, it is easy to check that $c_i^{n}$ is always less than $cp_i$ if $f_j \geq 0$. These ideas will be applied in the ascent method used later.

4.1 Dual Ascent Method for the Semi-Lagrangian Dual $SLD_\gamma$

In general, a dual ascent algorithm modifies a current value of multipliers in order to achieve a steady increase in $SLR(\gamma)$, see Bilde and Krarup [1977]. In our case this is possible due to the aforementioned properties. The method we used is explained under Algorithm 4.

Algorithm 4 Dual ascent method (DA) for $SLD_\gamma$.

Let $c_i^1 \leq \ldots \leq c_i^n$ be the sorted costs $c_{ij}$ and let $\gamma^0$ be an initial vector $\gamma$.

Step 1. (Initialization). Set $\epsilon > 0$, $iter = 0$. If $k > n$, $c_i^k = cp_i$.
For all $i \in I$ do:
- $cp_i = \max_j (c_{ij} + f_j)$,
- If $\gamma_i^0 < c_i^1$, then $\gamma_i^0 = c_i^1 + \epsilon$,
- Else,
  - If $\gamma_i^0 > c_i^n$, then $\gamma_i^0 = c_i^n$,
  - Else,
    - find a $j_i$ such that $\gamma_i$ belongs to $(c_i^j, c_i^{j+1})$, and set $\gamma_i^0 = c_i^j + \epsilon$.

Step 2. (Obtaining values $x_{ij}^{iter}$ and $y_{ij}^{iter}$). Solve $SLR(\gamma^{iter})$.

Step 3. (Finding a subgradient). Find a subgradient $s^{iter}$ of $SLR(\gamma^{iter})$.

Step 4. (Stop criterion). If $s^{iter} = 0$, STOP and $\gamma^{iter}$ is optimal. Otherwise, go to Step 5.

Step 5. (Updating multipliers) For each $i$ such that $s_i^{iter} = 1$, $j_i = j_i + 1$ and $\gamma_i^{iter+1} = \min \{c_i^j + \epsilon, cp_i\}$. Set $iter = iter + 1$. Go to step 2.

In Step 1, the algorithm gives an appropriate position of the terms of an initial multipliers vector in the intervals $I_j$. In Step 2, the procedure needs to solve a sequence of problems $SLR(\gamma)$, but due to the preference constraints, they are not as easy to solve as in the Lagrangian relaxation case $LR(\mu, \lambda)$ proposed before. However, it is possible to set some variables $x_{ij}$ equal to 0 in advance, in order to make it easier. Every $x_{ij}$ must be 0 if $c_{ij} - \gamma_i > 0$.

A subgradient for $SLR(\gamma)$ is computed in Step 3 as:

$$s = \left[ 1 - \sum_{j \in J} x_{ij}; \forall i \right] = [s, \gamma] \in \mathbb{R}^m,$$

where each component of $s$ belongs to $\{0, 1\}$ as the $\sum_{j \in J} x_{ij} \leq 1$ must be satisfied.

In Step 4, a stop criterion is given. Finally, the multipliers are updated (increased) by jumping from the current to the next $I_j$ interval for each component $i$ in $\gamma$. 

8
5 Speeding Up the Search for the Optimal Solution

After a certain number of iterations of DA that has been fixed beforehand, we apply a variable fixing heuristic VFH that takes, among all \( y_j = 1 \), a percentage of them by sorting this set of \( y_j \)'s by a determined criterion. Then, it sets all the selected \( y_j \) equal to 1. Finally the subproblem is solved. The method is described under Algorithm 5.

**Algorithm 5** Variable fixing heuristic (VFH) to speed up DA.

Let \( SLR_\gamma \) be an instance of DA after \( k \) iterations.

Let \( Y_\gamma \) be a set of \( y_j \)'s equal to 1 at instance \( SLR_\gamma \).

\[\text{Step 1. } \text{Set } ps \in [0, 1] \in \mathbb{R}. \text{ Sort the elements of } Y_\gamma \text{ such that } y_j < y_k \text{ if } \sum_i (c_{ij} + f_j) < \sum_i (c_{ik} + f_k).\]

\[\text{Step 2. } \text{Choose the first } ps \times 100\% \text{ smallest elements from the sorted } Y_\gamma \text{ to form the set } Y_{\gamma_{\text{subset}}}.\]

\[\text{Step 3. Set, } y_j = 1 \text{ for all } y_j \in Y_{\gamma_{\text{subset}}}.\]

\[\text{Step 4. Solve the original } SPLP.\]

Step 1 shows the criterion that we used to sort variables \( y_j \). We also tried to sort it by \( y_j < y_k \) if \( \sum_i p_{ij} < \sum_i p_{ik} \), where the \( p \)'s are the preferences. We obtained similar results, but the option proposed in Algorithm 5 was slightly better.

Another important issue is the starting point for the dual ascent method. We have tried two different starting \( \gamma \) vectors, \( \gamma = 0 \) and \( \gamma \) equal to \( \mu \), which is one of the multipliers that can be found using the subgradient method SG for \( LD_{\mu, \lambda} \) after a certain number of iterations. This \( \mu \) is the penalization associated with the relaxed family of constraints (1). We obtained better results with the second approach. This is because each sub-problem \( LR(\mu, \lambda) \) to be solved in the optimization of \( LD_{\mu, \lambda} \) is easy, see Theorem 1. For SG we used \( \mu_i^0 = \min_j \{ c_{ij} + f_j \} \) for all \( i \in I \) and \( \lambda_{ij} = 0 \) for all \( i \in I \) and \( j \in J \).

The whole accelerated dual ascent procedure is presented in Algorithm 6.

**Algorithm 6** Accelerated dual ascent algorithm. (ADA).

\[\text{Step 1. Set } \mu_i^0 = \min_j \{ c_{ij} + f_j \} \text{ for all } i \in I \text{ and } \lambda_{ij} = 0 \text{ for all } i \in I \text{ and } j \in J.\]

\[\text{Step 2. Run SG}(\mu_i^0, \lambda_i^0) \text{ during } sg_{\text{iter}} \text{ iterations. Find the iteration best}_{sg_{\text{iter}}} \text{ where is the best value of } LR(\mu, \lambda). \text{ Set } \gamma^0 = \mu_{\text{best}_{sg_{\text{iter}}}}.\]

\[\text{Step 3. During } da_{\text{iter}} \text{ iterations, run DA}(\gamma^0).\]

\[\text{Step 4. During } vfh_{\text{iter}} \text{ run DA. Get } Y_\gamma \text{ and run VFH. Save the solution.}\]

\[\text{Step 5. Find the best solution in the history and get best values for } x_{ij} \text{ and } y_j.\]

6 Computational results

In this section we present the computational results obtained after having applied the algorithms shown before. The experiments have been carried out on a PC with Intel® Xeon® 3.40 GHz processor and 16 Gb of RAM under a Windows® 7 operative system. All the procedures and algorithms have been written using the version 4.0.3 of the Mosel Xpress language and when a MIP problem needed to be solved we used FICO Xpress® version 8.0.

The instances were, at first, taken from Cánovas et al. (2006), which are based on the Beasley’s OR-Library (see Beasley, 1990). These are:

131, 132, 133, 134: \( m = 50 \) and \( n = 50 \)

a75_50, b75_50, c75_50: \( m = 75 \) and \( n = 50 \)
Additional data sets were generated using the same algorithm proposed in Cánovas et al. (2006):

- a100_75, b100_75, c100_75: $m = 100$ and $n = 75$
- a125_100, b125_100, c125_100: $m = 125$ and $n = 100$
- a150_100, b150_100, c150_100: $m = 150$ and $n = 100$

The headers of the tables have the following meanings:

- **Prob**: Name of the problem.
- **Opt**: Optimal objective function value of the problem P.
- **y_j**: Number of opened facilities.
- **bestUB**: Best upper bound.
- **GAP_o (%)**: bestUB-Opt ($\frac{\text{bestUB-Opt}}{\text{Opt}} \times 100\%$). The absolute and relative gap between the optimal and the value of an algorithm.
- **GAP_{LP}**: $\frac{\text{LP}(P) - \text{SG}}{\text{LP}(P)} \times 100\%$. The relative gap between the linear relaxation and the lower bound of the Lagrangian relaxation.
- **LP(P)**: Linear relaxation value for problem P.
- **SG**: Best value using the subgradient method for $LD_{\mu \lambda}$.
- **iter (itbsol)**: Number of iterations (iteration where best solution was found).
- **t (Tt)**: CPU time in seconds (Total time).
- **sol10%**: First solution found that is within 10% or less of the optimal solution.
- **UB (LB)**: Upper bound (Lower bound).

First, we show in Table 1 the results for Algorithms Hs and Hc. As can be seen, we obtain better results with the second one. It finds the optimal solution for 6 of the first 16 problems and in the rest of the instances it is not too far from the optimal with an average gap of 4.65%. All runs performed took less than 1 second whereas Xpress needs between 3 and 12 to solve these instances to optimality. The value obtained by Hc was used as an upper bound for the SG method, Algorithm 1.

Table 2 shows the performance of the subgradient method SG (Algorithm 1) over the first 40 data sets. The algorithm was stopped when the parameter $\beta$ was equal to 0 with a starting value of 2, and it was decreasing linearly at a rate of 0.005 if a solution did not change after 30 iterations. We also show the results for the first solution within a 10% from $\text{LP}(P)$.

After the time reported, SG did not obtain the value $\text{LP}(P)$ for any of the problems. However, as mentioned before, the SG will be useful to find initial multipliers for the dual ascend DA.

Finally, we can see the results for the algorithm ADA in Table 3. The routine needs parameters $sg_{iter}$, $da_{iter}$ and $vfh_{iter}$, and different values were used for the four groups of data sets tested. These were empirically set as:

- a75_50, b75_50, c75_50: $sg_{iter} = 50$, $da_{iter} = 3$ and $vfh_{iter} = 2$
- a100_75, b100_75, c100_75: $sg_{iter} = 100$, $da_{iter} = 7$ and $vfh_{iter} = 2$
- a125_100, b125_100, c125_100: $sg_{iter} = 170$, $da_{iter} = 10$ and $vfh_{iter} = 2$
- a150_100, b150_100, c150_100: $sg_{iter} = 170$, $da_{iter} = 12$ and $vfh_{iter} = 2$

and for all the instances $ps = 0.25$ (see Algorithm 5).
| Prob | Opt | y_j | bestUB | GAP | GAP_o | GAP_o% | y_j | bestUB | GAP | GAP_o | GAP_o% | y_j |
|------|-----|-----|--------|-----|-------|--------|-----|--------|-----|-------|--------|-----|
| 131_1 | 1001440 | 6 | 1001440 | 0 | 0.00% | 6 | 1001440 | 0 | 0.00% | 6 |
| 131_2 | 982517  | 9 | 1003100 | 20583 | 2.09% | 5 | 982517  | 0 | 0.00% | 9 |
| 131_3 | 1038953 | 10 | 1248143 | 208290 | 20.03% | 1 | 1139012 | 99159 | 9.54% | 9 |
| 131_4 | 1028447 | 9 | 1248143 | 219696 | 21.36% | 1 | 1049791 | 21344 | 2.08% | 7 |
| 132_1 | 1122750 | 8 | 1248143 | 125393 | 11.17% | 1 | 1239961 | 117211 | 10.44% | 7 |
| 132_2 | 1157722 | 9 | 1248143 | 90421 | 7.81% | 1 | 1199057 | 41335 | 3.57% | 9 |
| 132_3 | 1146301 | 6 | 1248143 | 101842 | 8.88% | 1 | 1146301 | 0 | 0.00% | 6 |
| 132_4 | 1036779 | 5 | 1248143 | 211364 | 20.39% | 1 | 1036779 | 0 | 0.00% | 5 |
| 133_1 | 1103272 | 7 | 1248143 | 144871 | 13.13% | 1 | 1103272 | 0 | 0.00% | 7 |
| 133_2 | 1035443 | 5 | 1248143 | 212700 | 20.54% | 1 | 1100713 | 65270 | 6.30% | 7 |
| 133_3 | 1171331 | 6 | 1248143 | 76812 | 6.56% | 1 | 1208198 | 36867 | 3.15% | 4 |
| 133_4 | 1083636 | 9 | 1248143 | 164507 | 15.18% | 1 | 1090582 | 6946  | 0.64% | 9 |
| 134_1 | 1179639 | 4 | 1248143 | 68504 | 5.81% | 1 | 1179639 | 0 | 0.00% | 4 |
| 134_2 | 1121633 | 7 | 1248143 | 126510 | 11.28% | 1 | 1205809 | 84176 | 7.50% | 5 |
| 134_3 | 1171409 | 6 | 1248143 | 76734 | 6.55% | 1 | 1173693 | 2284  | 0.19% | 7 |
| 134_4 | 1210465 | 3 | 1248143 | 37678 | 3.11% | 1 | 1248143 | 37678 | 3.11% | 1 |
| a75_50_1 | 1661269 | 7 | 1787955 | 126686 | 7.63% | 1 | 1787955 | 126686 | 7.63% | 1 |
| a75_50_2 | 1632907 | 6 | 1784848 | 151941 | 9.30% | 2 | 1775976 | 146669 | 8.98% | 4 |
| a75_50_3 | 1632213 | 7 | 1738404 | 106191 | 6.51% | 3 | 1738404 | 106191 | 6.51% | 3 |
| a75_50_4 | 1585028 | 5 | 1787955 | 202927 | 12.80% | 1 | 1709978 | 124950 | 7.88% | 5 |
| b75_50_1 | 1252804 | 8 | 1374685 | 76812 | 6.56% | 1 | 1280198 | 36867 | 3.15% | 4 |
| b75_50_2 | 1337446 | 9 | 1403629 | 66183 | 4.95% | 5 | 1403629 | 66183 | 4.95% | 5 |
| b75_50_3 | 1201706 | 12 | 1386817 | 185111 | 15.40% | 13 | 1386817 | 185111 | 15.40% | 13 |
| b75_50_4 | 1334782 | 11 | 1440836 | 37678 | 3.11% | 1 | 1248143 | 37678 | 3.11% | 1 |
| a100_75_1 | 2286397 | 4 | 2476632 | 190235 | 8.32% | 1 | 2459349 | 172952 | 7.56% | 4 |
| a100_75_2 | 2463187 | 3 | 2476632 | 13445 | 0.55% | 1 | 2476632 | 13445 | 0.55% | 1 |
| a100_75_3 | 2415836 | 3 | 2476632 | 60796 | 2.52% | 1 | 2467003 | 51167 | 2.12% | 3 |
| a100_75_4 | 2380150 | 4 | 2476632 | 96482 | 4.05% | 1 | 2476632 | 96482 | 4.05% | 1 |
| b100_75_1 | 1950231 | 8 | 2061201 | 110970 | 5.69% | 7 | 2061201 | 110970 | 5.69% | 7 |
| b100_75_2 | 2030997 | 8 | 2389395 | 366298 | 18.11% | 1 | 2180286 | 157189 | 7.77% | 8 |
| b100_75_3 | 2062595 | 8 | 2133724 | 71129 | 3.45% | 8 | 2133724 | 71129 | 3.45% | 8 |
| b100_75_4 | 1865323 | 9 | 1994265 | 128942 | 6.91% | 7 | 1994265 | 128942 | 6.91% | 7 |
| c100_75_1 | 1843620 | 6 | 2107973 | 264353 | 14.34% | 5 | 2090221 | 246601 | 13.38% | 7 |
| c100_75_2 | 1808867 | 11 | 2025331 | 216464 | 11.97% | 9 | 2005071 | 196204 | 10.85% | 13 |
| c100_75_3 | 1820587 | 8 | 2019651 | 199064 | 10.93% | 7 | 2019651 | 199064 | 10.93% | 7 |
| c100_75_4 | 1839007 | 9 | 2046525 | 207518 | 11.28% | 8 | 2046525 | 207518 | 11.28% | 8 |
Table 2: Computational results of subgradient method.

| Prob | LP(P) | SG | itbsol | iter | t | GAP_{LP} | sol10% | iter | t | GAP_{LP} |
|------|-------|----|--------|------|---|-----------|--------|------|---|-----------|
| 13_{1.2} | 925492 | 881876 | 1474 | 1500 | 19 | 4.71% | 833518 | 613 | 8 | 9.00% |
| 13_{1.2} | 925195 | 872136 | 1450 | 1500 | 19 | 5.73% | 833747 | 695 | 9 | 9.00% |
| 13_{1.2} | 955447 | 929854 | 1488 | 1500 | 20 | 2.68% | 860178 | 636 | 8 | 10.00% |
| 13_{1.2} | 933025 | 903486 | 1462 | 1500 | 19 | 3.17% | 841581 | 580 | 7 | 9.80% |
| 13_{2.2} | 1007417 | 981624 | 1500 | 1500 | 19 | 2.56% | 907222 | 479 | 6 | 9.90% |
| 13_{2.2} | 990513 | 957530 | 1389 | 1425 | 18 | 3.33% | 897874 | 445 | 6 | 9.40% |
| 13_{2.2} | 955447 | 929854 | 1488 | 1500 | 20 | 2.68% | 860178 | 636 | 8 | 10.00% |
| 13_{2.2} | 933025 | 903486 | 1462 | 1500 | 19 | 3.17% | 841581 | 580 | 7 | 9.80% |
| 13_{3.2} | 1007417 | 981624 | 1500 | 1500 | 19 | 2.56% | 907222 | 479 | 6 | 9.90% |
| 13_{3.2} | 990513 | 957530 | 1389 | 1425 | 18 | 3.33% | 897874 | 445 | 6 | 9.40% |
| 13_{3.2} | 955447 | 929854 | 1488 | 1500 | 20 | 2.68% | 860178 | 636 | 8 | 10.00% |
| 13_{3.2} | 933025 | 903486 | 1462 | 1500 | 19 | 3.17% | 841581 | 580 | 7 | 9.80% |
| 13_{4.2} | 1007417 | 981624 | 1500 | 1500 | 19 | 2.56% | 907222 | 479 | 6 | 9.90% |
| 13_{4.2} | 990513 | 957530 | 1389 | 1425 | 18 | 3.33% | 897874 | 445 | 6 | 9.40% |
| 13_{4.2} | 955447 | 929854 | 1488 | 1500 | 20 | 2.68% | 860178 | 636 | 8 | 10.00% |
| 13_{4.2} | 933025 | 903486 | 1462 | 1500 | 19 | 3.17% | 841581 | 580 | 7 | 9.80% |
| 13_{5.2} | 1007417 | 981624 | 1500 | 1500 | 19 | 2.56% | 907222 | 479 | 6 | 9.90% |
| 13_{5.2} | 990513 | 957530 | 1389 | 1425 | 18 | 3.33% | 897874 | 445 | 6 | 9.40% |
| 13_{5.2} | 955447 | 929854 | 1488 | 1500 | 20 | 2.68% | 860178 | 636 | 8 | 10.00% |
| 13_{5.2} | 933025 | 903486 | 1462 | 1500 | 19 | 3.17% | 841581 | 580 | 7 | 9.80% |
| 13_{6.2} | 1007417 | 981624 | 1500 | 1500 | 19 | 2.56% | 907222 | 479 | 6 | 9.90% |
| 13_{6.2} | 990513 | 957530 | 1389 | 1425 | 18 | 3.33% | 897874 | 445 | 6 | 9.40% |
| 13_{6.2} | 955447 | 929854 | 1488 | 1500 | 20 | 2.68% | 860178 | 636 | 8 | 10.00% |
| 13_{6.2} | 933025 | 903486 | 1462 | 1500 | 19 | 3.17% | 841581 | 580 | 7 | 9.80% |

...
Table 3: Computational results for ADA.

| Prob | Optimal Xpress | Hc | SG | DA with VFH | ADA |
|------|----------------|----|----|-------------|-----|
|      |                |    |    |             |     |
| a75_50_1 | 166.1269 1201542 | 7  | 16 | 1787955 1 0 | 832797 4 |
| a75_50_2 | 163.2907 1188359 | 6  | 21 | 1779576 4 0 | 828462 4 |
| a75_50_3 | 163.2213 1189183 | 7  | 18 | 1738404 3 0 | 783044 4 |
| a75_50_4 | 158.5028 1200006 | 5  | 18 | 1709978 5 0 | 863178 3 |
| b75_50_1 | 125.2804 900017  | 8  | 11 | 1343201 10 0 | 355739 2 |
| b75_50_2 | 133.7446 922048  | 9  | 20 | 1403629 5 0 | 376669 3 |
| b75_50_3 | 124.9750 897073  | 9  | 17 | 1368788 8 0 | 348636 3 |
| b75_50_4 | 121.7508 885392  | 9  | 12 | 1438203 10 0 | 414596 3 |
| c75_50_1 | 131.0193 892387  | 11 | 17 | 1375866 11 0 | 550106 4 |
| c75_50_2 | 124.4255 882933  | 10 | 11 | 1316595 11 0 | 533586 3 |
| c75_50_3 | 120.1706 895951  | 12 | 8  | 1386817 13 0 | 549787 3 |
| c75_50_4 | 133.4757 929064  | 11 | 21 | 1440836 5 0 | 581078 4 |
| a100_75_1 | 2286397 1800032 | 4  | 61 | 2459349 4 1 | 1649771 16 |
| a100_75_2 | 2463187 1793377  | 13 | 1 | 2476632 16 | 1982594 3 |
| a100_75_3 | 2415836 1788393  | 14 | 148 | 2467003 3 1 | 1634629 3 |
| a100_75_4 | 2380150 1795298  | 14 | 124 | 2476632 3 1 | 1628551 3 |
| b100_75_1 | 1952031 1330138  | 8  | 624 | 2061201 7 1 | 1092887 17 |
| b100_75_2 | 2023097 1353824  | 8  | 727 | 2182086 8 1 | 1150334 18 |
| b100_75_3 | 2062595 1351603  | 8  | 841 | 2133724 8 1 | 1165120 18 |
| b100_75_4 | 1865323 1332525  | 9  | 546 | 1994265 7 1 | 1143446 15 |
| c100_75_1 | 1843620 1250876  | 6  | 430 | 2090221 7 1 | 1077180 20 |
| c100_75_2 | 1808867 1239182  | 11 | 396 | 2005071 13 0 | 1053868 19 |
| c100_75_3 | 1820587 1232462  | 8  | 423 | 2019651 7 1 | 1054526 16 |
| c100_75_4 | 1839007 1243861  | 9  | 583 | 2046525 8 1 | 1064170 21 |
| a125_100_1 | 3041451 2392412  | 2  | 257 | 3070535 1 1 | 2235753 47 |
| a125_100_2 | 3040248 2393448  | 2  | 302 | 3070535 1 1 | 2232859 48 |
| a125_100_3 | 3055260 2362216  | 3  | 325 | 3070535 1 1 | 2227182 47 |
| a125_100_4 | 3056428 2381167  | 2  | 334 | 3070535 1 1 | 2244017 54 |

Continued on next page
Table 3: Computational results for ADA.

| Prob  | Xpress | Optimal | LP  | y_i | t | bestUB | y_i | t | LB  | bestUB | LB  | y_i | t | Tt | imp | GAP, % |
|-------|--------|---------|-----|-----|---|--------|-----|---|-----|--------|-----|-----|---|----|-----|--------|
| b125_100_1 | 2640798 | 1794710 | 7 | 5297 | 2850664 | 5 | 1 | 1601985 | 58 | 2640798 | 2082726 | 7 | 989 | 1047 | 80% | 0.00% |
| b125_100_2 | 2550592 | 1790859 | 7 | 2086 | 2808259 | 9 | 1 | 1600582 | 56 | 2568522 | 2046667 | 7 | 848 | 903 | 57% | 0.70% |
| b125_100_3 | 2604906 | 1792133 | 4 | 4059 | 2782609 | 5 | 1 | 1607076 | 58 | 2604906 | 1977647 | 4 | 435 | 494 | 88% | 0.00% |
| b125_100_4 | 2580595 | 1792581 | 7 | 2686 | 2778713 | 4 | 1 | 1587315 | 60 | 2637611 | 1975516 | 7 | 358 | 419 | 84% | 2.21% |
| c125_100_1 | 2491714 | 1663455 | 10 | 7805 | 2669965 | 9 | 1 | 1491505 | 61 | 2491714 | 1953129 | 10 | 1685 | 1746 | 78% | 0.00% |
| c125_100_2 | 2468480 | 1674102 | 9 | 6296 | 2674261 | 8 | 1 | 1487416 | 58 | 2518055 | 1957245 | 10 | 519 | 577 | 91% | 2.01% |
| c125_100_3 | 2559381 | 1672005 | 8 | 11381 | 2685807 | 7 | 1 | 1496552 | 59 | 2559381 | 1957575 | 8 | 1177 | 1236 | 89% | 0.00% |
| c125_100_4 | 2538550 | 1691148 | 8 | 6407 | 2683676 | 9 | 1 | 1503317 | 61 | 2561098 | 2016969 | 8 | 726 | 786 | 88% | 0.89% |
| a150_100_1 | 3768087 | 2906284 | 1 | 371 | 3768087 | 1 | 2 | 2691219 | 57 | 3768087 | 3200250 | 3 | 338 | 395 | -6% | 0.00% |
| a150_100_2 | 3768087 | 2891681 | 1 | 457 | 3768087 | 1 | 1 | 2695660 | 56 | 3768087 | 3212545 | 2 | 329 | 384 | 16% | 0.00% |
| a150_100_3 | 3741364 | 2882917 | 3 | 340 | 3768087 | 1 | 1 | 2691438 | 58 | 3741364 | 3185719 | 3 | 306 | 363 | -7% | 0.00% |
| a150_100_4 | 3768087 | 2911017 | 1 | 591 | 3768087 | 1 | 1 | 2697003 | 63 | 3768087 | 3241759 | 2 | 337 | 400 | 32% | 0.00% |
| b150_100_1 | 3271859 | 2159537 | 4 | 15483 | 3487710 | 1 | 4 | 1916485 | 68 | 3271859 | 2651812 | 4 | 2357 | 2426 | 84% | 0.00% |
| b150_100_2 | 3227897 | 2160753 | 5 | 9677 | 3457623 | 5 | 2 | 1935777 | 68 | 3227897 | 2546299 | 5 | 975 | 1043 | 89% | 0.00% |
| b150_100_3 | 3100755 | 2148658 | 6 | 5986 | 3390435 | 8 | 1 | 1920876 | 69 | 3150075 | 2582861 | 6 | 1081 | 1150 | 81% | 0.00% |
| b150_100_4 | 3342783 | 2109488 | 5 | 9596 | 3657438 | 8 | 1 | 1927555 | 69 | 3342783 | 2604144 | 5 | 2225 | 2294 | 76% | 0.00% |
| c150_100_1 | 2979389 | 1988908 | 9 | 10008 | 3198861 | 5 | 1 | 1790894 | 71 | 2979389 | 2483032 | 9 | 1520 | 1591 | 84% | 0.00% |
| c150_100_2 | 3109105 | 1985389 | 7 | 21734 | 3291990 | 6 | 2 | 1766096 | 72 | 3109105 | 2404613 | 7 | 949 | 1021 | 95% | 0.00% |
| c150_100_3 | 2937767 | 1982388 | 8 | 10590 | 3079664 | 4 | 1 | 1770318 | 71 | 2954519 | 2379867 | 7 | 2369 | 2440 | 77% | 0.57% |
| c150_100_4 | 3165327 | 1997014 | 10 | 65134 | 3189247 | 7 | 1 | 1775186 | 75 | 3176587 | 2429997 | 12 | 5842 | 5917 | 91% | 0.36% |
With the exception of the first group \((m = 75, n = 50)\) and a few instances in the rest, the times obtained by ADA when it meets the optimal solution are considerably lower than those obtained by Xpress. It can be observed particularly in the last group. For example, the instance \(c_{150,100,2}\) had a reduction of 95\% on the total time. ADA is always close to the optimal values, in fact, the GAP average is only 0.43\%.

### 7 Conclusions

In this paper we have proposed a heuristic method to solve the SPLPO inspired by techniques introduced in \cite{Cornuejols:1977, Beltran:2012}. The assignment and VUBs constraints in SPLPO were relaxed to formulate a Lagrangean problem. We solved its dual with a subgradient method and used a vector of multipliers as a starting point of an ascent algorithm for the dual of a semi-Lagrangean formulation. Nevertheless, a better starting point should be the multipliers obtained by relaxing only the assignment constraints, the same family of constraints relaxed in our proposed SPLPO semi-Lagrangean problem. However, the sequence of linear subproblems that need to be solved in the subgradient method are not as easy to solve as those when our proposed relaxation is used. We used the variable fixing heuristic VFH because MIP problems in ADA are becoming increasingly difficult as the number of iterations grows. We have shown that the ADA algorithm works particularly well on large instances, but there must be a future discussion about the parameters settings.

### 8 Acknowledgements

This work has been supported in part by SENESCYT-Ecuador (National Secretary of Higher Education, Science, Technology and Innovation of Equator). The research of Sergio García has been funded by Fundación Séneca (project 19320/PI/14).

### References

Beasley, E. (1990). OR-library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072.

Beltrán, C., Tandoki, C., and Vial, J. (2006). Solving the p-median problem with a semi-Lagrangian relaxation. *Computational Optimization and Applications*, 35:239–260.

Beltrán, C., Vial, J., and Alonso, A. (2012). Semi-Lagrangian relaxation applied to the uncapacitated facility location problem. *Computational Optimization and Applications*, 51:387–409.

Bilde, O. and Krarup, J. (1977). Sharp lower bounds and efficient algorithms for the simple plant location problem. *Annals of Discrete Mathematics*, 1:79–97.

Cánovas, L., García, S., Labbé, M., and Marín, A. (2006). A strengthened formulation for the simple plant location problem with order. *Operations Research Letters*, 35:141–150.

Conforti, M., Cornuéjols, G., and Zambelli, G. (2014). *Integer programming*. Springer.

Cornuéjols, G., Fisher, M., and Nemhauser, G. (1977). Location of bank accounts to optimize float: an analytic study of axact and approximated algorithms. *Management Science*, 23(8):789–810.

Fisher, M. L. (2004). The Lagrangian relaxation method for solving integer programming problems algorithms. *Management Science*, 50(12):1861–1871.

Geoffrion, A. (1974). Lagrangian relaxation for integer programming. *Mathematical Programming Study*, 2:82–114.
Guignard, M. (2003). Lagrangean relaxation. A tutorial. *TOP*, 11(2):151–228.

Held, M. and Karp, R. (1971). The traveling salesman problem and minimum spanning trees: part II. *Mathematical Programming*, 1:6–25.

Held, M., Wolfe, P., and Crowder, H. (1974). Validation of subgradient optimization. *Mathematical Programming*, 6:62–88.

Jörnsten, K. (2016). An improved Lagrangian relaxation and dual ascent approach to facility location problems. *Computational Management Science*, 13:317–348.

Monabebati, E. (2014). An application of a Lagrangian-type relaxation for the uncapacitated facility location problem. *Japan Journal of Industrial and Applied Mathematics*, 31:483–499.

Poljak, B. T. (1967). A general method for solving extremum problems. *Soviet Mathematics Doklady*, 8:593–597.

Vasilyev, I. and Klimentova, X. (2010). The branch and cut method for the facility location problem with client’s preferences. *Journal of Applied and Industrial Mathematics*, 4(3):441–454.

Vasilyev, I. L., Klimentova, X., and Boccia, M. (2013). Polyhedral study of simple plant location problem with order. *Operations Research Letters*, 41:153–158.