Improving the Semi-Lagrangian Relaxation Approach to the Simple Plant Location Problem

Lauren Durrell[*] Thu H. Dang[†] Adam N. Letchford[†]

January 2025; Revised June 2025, November 2025

Abstract

The *Simple Plant Location Problem* (SPLP) is a much-studied combinatorial optimisation problem with many applications. In 2012, Beltran-Royo *et al.* devised a promising exact algorithm for the SPLP. Their approach is based on "semi-Lagrangian relaxation", which is known to be effective for integer programs with equality constraints. We show that one can speed up their algorithm by (a) using a fast method to initialise the Lagrangian multipliers, and (b) eliminating redundant variables from the relaxed problem. Computational experiments, on several sets of benchmark instances, show that the improved algorithm consistently either reaches the optimal value much more quickly or, within the same time limit, produces tighter lower bounds with fewer iterations.

Keywords: facility location; combinatorial optimisation; semi-Lagrangian relaxation

1 Introduction

The *Simple Plant Location Problem* (SPLP), also called the *Uncapacitated Facility Location Problem*, is a classic NP-hard combinatorial optimisation problem [22] and an important problem in location science [25]. In the SPLP, we are given m facilities and n clients. The cost of opening facility i is f_i, and the cost of assigning client j to facility i is c_{ij}. The task is to decide which facilities to open, and assign each client to an open facility, while minimizing the total cost.

Numerous exact and heuristic algorithms have been proposed for the SPLP (see the surveys [10, 22, 24, 30]). At the time of writing, the most promising exact algorithms are those described in [6, 11, 14, 27, 29]. The

[*]STOR-i Centre for Doctoral Training, Lancaster University, Lancaster LA1 4YR, UK. E-mail: `L.Durrell@lancaster.ac.uk`

[†]Department of Management Science, Lancaster University, Lancaster LA1 4YX, UK. E-mail: `{T.H.Dang,A.N.Letchford}@lancaster.ac.uk`

algorithms in [11, 14] are based on Benders decomposition, whereas the ones in [27] and [29] are based on aggressive reduction and primal-dual heuristics, respectively. Here, we focus on the approach of Beltran-Royo, Vial and Alonso-Ayuso [6], which we call the BVA approach for short.

The BVA approach is based on *semi-Lagrangian relaxation*, or SLR for short. SLR is a variant of Lagrangian relaxation that is suitable for integer programs with equality constraints [5]. The idea is to replace each equality with two inequalities, and then relax only one of the inequalities in Lagrangian fashion. Although the idea is fairly simple, it exhibits strong duality and has been applied with great success to several combinatorial optimisation problems (e.g., [4, 5, 6, 8, 31]).

In this article, we examine the BVA approach in more detail. We begin by presenting a fast and effective method for computing the initial values of the (semi-)Lagrangian multipliers. We then prove two theoretical results, which enable one to solve the relaxed problem more quickly. Finally, we conduct extensive computational experiments on benchmark instances, comparing the original BVA approach with our improved version. The results indicate that our approach is capable of solving more SPLP instances to proven optimality in acceptable computing times.

Throughout the paper, we will often refer to the following integer programming (IP) formulation of the SPLP, due to Balinski [1]:

$$\begin{array}{lll} \min & \sum_{i=1}^m f_i y_i + \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} & & (1) \\ \text{s.t.} & \sum_{i=1}^m x_{ij} = 1 & (j = 1, \ldots, n) & (2) \\ & x_{ij} \le y_i & (i = 1, \ldots, m;\ j = 1, \ldots, n) & (3) \\ & y \in \{0,1\}^m, x \in \{0,1\}^{m \times n}. & & (4) \end{array}$$

Here, the binary variable y_i takes the value 1 if and only if facility i is opened, and the binary variable x_{ij} takes the value 1 if and only if client j is assigned to facility i. The constraints (2) ensure that each client is assigned to a facility, and the constraints (3) ensure that clients are assigned only to open facilities.

We will also let c_j^k denote the k-th smallest value of c_{ij} for a given client j. We allow repeated values. For example, if $m = 4$ and $c_{1j} = c_{3j} = 80$, $c_{2j} = 50$ and $c_{4j} = 100$, then $c_j^1 = 50$, $c_j^2 = c_j^3 = 80$ and $c_j^4 = 100$.

The paper is organised as follows. Section 2 is a literature review. Section 3 describes our procedures for accelerating the BVA approach. Section 4 contains the computational results, and Section 5 offers some suggestions for future research.

2 Literature Review

Since the literature on the SPLP is huge, we refer here only to works of direct relevance. The following three subsections cover dual ascent, Lagrangian relaxation and SLR, respectively.

2.1 Dual ascent

In practice, solving the continuous relaxation of the IP (1)-(4) can be very time-consuming for large SPLP instances. A much faster way to compute lower bounds is given by *dual ascent* [7, 13]. We begin by taking the dual of the continuous relaxation, which is:

$$\begin{array}{lll} \max & \sum_{j=1}^n v_j & \\ \text{s.t.} & \sum_{j=1}^n w_{ij} \le f_i & (\forall i) \\ & v_j - w_{ij} \le c_{ij} & (\forall i, j) \\ & v \in \mathbb{R}^n, w \in \mathbb{R}_+^{m \times n}. & \end{array}$$

Here, the v_j and w_{ij} are the dual variables for (2) and (3), respectively.

It is then clear that there is an optimal solution to the dual such that $w_{ij} = \max\{0, v_j - c_{ij}\}$ for all i and j. Thus, we can eliminate the w variables to obtain the so-called 'condensed' dual:

$$\begin{array}{lll} \max & \sum_{j=1}^n v_j & \quad (5) \\ \text{s.t.} & \sum_{j=1}^n \max\{0, v_j - c_{ij}\} \le f_i \quad (\forall i) & \quad (6) \\ & v \in \mathbb{R}^n. & \quad (7) \end{array}$$

We then solve the condensed dual heuristically by setting each v_j to c_j^1 and then iteratively increasing one v_j at a time as long as possible.

Dual ascent can be used to produce upper bounds as well as lower bounds. For details, see [7, 13]. Also see [19, 21, 26] for further improvements.

2.2 Lagrangian relaxation

Another way to compute lower bounds for the SPLP is *Lagrangian relaxation* [3, 15]. We relax the constraints (2), with a vector $\lambda \in \mathbb{R}_+^n$ of Lagrangian multipliers, to yield the relaxed problem:

$$\sum_{j=1}^n \lambda_j + \min\left\{ \sum_{i=1}^m f_i y_i + \sum_{i=1}^m \sum_{j=1}^n (c_{ij} - \lambda_j) x_{ij} : (3), (4) \right\}. \tag{8}$$

This problem decomposes into m subproblems. The subproblem for a given facility i takes the form:

$$\min\left\{ f_i y_i + \sum_{j=1}^n (c_{ij} - \lambda_j) x_{ij} : x_{ij} \le y_i \ (\forall j),\ y_i \in \{0,1\},\ x_{ij} \in \{0,1\} \ (\forall j) \right\}.$$

To solve this subproblem, compute the following 'Lagrangian reduced cost' of facility i:

$$\bar{f}_i = f_i - \sum_{j=1}^{m} \max\big\{0, \lambda_j - c_{ij}\big\}. \tag{9}$$

Then, if $\bar{f}_i < 0$, set y_i to 1 and set x_{ij} to 1 for all clients j such that $c_{ij} < \lambda_j$. Otherwise, set y_i to 0 and set all associated x variables to 0.

To find good Lagrangian multipliers, the authors of [3, 15] simply use the subgradient method. They also call a primal heuristic after each subgradient iteration.

2.3 Semi-Lagrangian relaxation

As mentioned above, SLR was applied to the SPLP in [6]. The idea is to split the equations (2) into

$$\sum_{i=1}^{m} x_{ij} \leq 1 \qquad (\forall j). \tag{10}$$

and

$$\sum_{i=1}^{m} x_{ij} \geq 1 \qquad (\forall j). \tag{11}$$

The latter inequalities are then relaxed in Lagrangian fashion, and the former inequalities are retained in the relaxed problem.

The inclusion of (10) in the relaxation makes it harder to solve, since it no longer decomposes into m independent subproblems. On the other hand, it makes the lower bound stronger. In fact, there exists a set of multipliers λ_j, each lying within the half-open interval $(\min_i\{c_{ij}\},\ \min_i\{c_{ij}+f_i\}]$, such that the gap between the lower bound and the optimal value is completely closed [6]. This means in particular that the relaxation exhibits strong duality.

Note that we can eliminate the variable x_{ij} from the relaxed problem if $c_{ij} - \lambda_j \geq 0$. This led the authors of [6] to construct a certain graph. For a given λ, define the bipartite graph $G_\lambda = \big(V, E_\lambda\big)$, where V has one node for each facility and each client, and the edge $\{i, j\}$ is present if and only if $c_{ij} < \lambda_j$. If G_λ is not connected, then the relaxed problem for the given λ decomposes into smaller subproblems, one for each non-trivial connected component of G_λ (where 'non-trivial' means containing at least one edge).

Based on the above, the authors of [6] proposed Algorithm 1, which solves the SPLP exactly. Note that the multipliers are computed in an iterative manner, similar to dual ascent. The initial choice for the integers $\ell(j)$ will be discussed in Subsection 3.1.

SLR has been applied successfully to several other location problems. Beltran-Royo *et al.* [5] applied it to the p-median problem, and obtained good results for instances with up to 4000 nodes. Zhang et al. [32] applied it to a

Algorithm 1: Simplified BVA Approach to the SPLP

```
   input  : number of facilities $m$, number of clients $n$,
            facility costs $f_i$, assignment costs $c_{ij}$, positive constant $\epsilon$,
            integer $\ell(j) \in \{1, \dots, m\}$ for each client $j$.
 1 for each client $j$ do
 2     Compute and store the $c_j^k$ values;
 3     Set $\tilde{c}_j$ to $\min\{c_{ij} + f_i\}$ and $c_j^{m+1}$ to $\infty$;
 4     Set $\lambda_j$ to $c_j^{\ell(j)} + \epsilon$;
 5 end
 6 Set Stop to false;
 7 while Stop $=$ false do
 8     Construct the graph $G_\lambda$ and compute the connected components;
 9     for each non-trivial component of $G_\lambda$ do
10         Solve the corresponding subproblem using any IP solver;
11     end
12     Let $(x^*, y^*)$ be the solution of the relaxed problem;
13     for each client $j$ do
14         if client $j$ is not currently assigned to a facility then
15             Increase $\lambda_j$ to $\min\left\{c_j^{\ell(j)+1}, \tilde{c}_j\right\} + \epsilon$;
16             Increase $\ell(j)$ to $\min\left\{\ell(j) + 1, m\right\}$;
17         end
18     end
19     if all clients are assigned to a facility then
20         Stop $=$ true;
21     end
22 end
   output : optimal solution $(x^*, y^*)$
```

problem involving the location of distribution centres for B2C E-Commerce, and showed that the method generally outperformed CPLEX. Cabezas and co-authors [9, 8] applied SLR to variants of the SPLP that incorporate customer preferences. In both papers, they were able to obtain strong bounds quickly for instances with up to 150 facilities and 100 clients.

SLR has also been applied to other problems, besides location problems. Zhang *et al.* [31] applied it to the Quadratic Assignment Problem and Belik and Jörnsten [4] applied it to the k-Cardinality Assignment Problem. For the sake of brevity, we do not go into the details.

To close this section, we remark that a simplified SLR approach to the SPLP was proposed in [28] and further studied in [4] and [20]. The idea is to aggregate the constraints (11) into a single constraint, and have just one Lagrangian multiplier. Again, we omit details for brevity.

3 Accelerating the BVA Approach

In this section, we show how to accelerate the BVA approach in several ways. In Subsection 3.1, we present a fast way to find the initial set of multipliers. In Subsections 3.2 and 3.3, we prove two results which enable us to reduce the number of variables in the relaxed problem, for a given set of multipliers. In Subsection 3.4, we give a worked example.

Throughout this section, we let S_j denote $\{c_{1j}, \ldots, c_{mj}\}$. Using the same example as in the introduction, if $m = 4$ and $c_{1j} = c_{3j} = 80$, $c_{2j} = 50$ and $c_{4j} = 100$, then $S_j = \{50, 80, 100\}$.

3.1 Finding the initial multiplier values

Note that Algorithm 1 requires as input an initial $\ell(j)$ value for each client j. One can, of course, just set all of those values to 1, but this makes the algorithm unnecessarily slow. Two alternative ways to determine the values were discussed in [6]:

- Solve the continuous relaxation of the IP (1)-(4), and let v^* be the vector of dual prices for the constraints (2). For each client j, let α_j be the value in S_j that is closest to v_j^*, and let $\ell(j)$ be the largest integer such that $c_j^{\ell(j)} = \alpha_j$.
- Run a standard Lagrangian relaxation algorithm, using the subgradient method, to obtain an initial vector λ'. For each client j, let α_j be the value in S_j that is closest to λ'_j, and proceed as above.

In our experience, both of these options are very time-consuming for large instances. To address this, we use a result from [26]. Consider once more the dual LP (5)-(7). Let us say that the vector v is 'at level k' if $v_j = c_j^k$ for all j. Also, let us define the 'base level' as the largest integer, say b, such that we can set v to level b, while still satisfying (6). It is shown in [26] that we can compute the base level in $O(nb \log b)$ time. We have found that, in practice, the algorithm in [26] is extremely fast.

Our proposal, then, is to compute the base level b, using the algorithm in [26], before running Algorithm 1. We then set all of the $\ell(j)$ to b in the input to Algorithm 1.

Note that, by the definition of the base level, together with constraints (6), we have

$$f_i - \sum_{j=1}^{m} \max\left\{0, c_j^b - c_{ij}\right\} \geq 0 \qquad (\forall i).$$

This means that, after initialising λ in the way we have described, the 'Lagrangian reduced cost' of each facility, called $\bar{f}_i$ in (9), is either non-negative or close to zero for all i. We have found that this initialisation provides an excellent starting point, substantially reducing the number of iterations of the 'while' loop in Algorithm 1.

3.2 Eliminating facilities from the relaxed problem

Before presenting our second idea, we remind the reader that, if we ignore the constant term $\sum_{j=1}^{n} \lambda_j$, then the relaxed problem in SLR takes the form:

$$\begin{array}{llr} & \min \sum_{i=1}^{m} f_i y_i + \sum_{i=1}^{m} \sum_{j=1}^{n} (c_{ij} - \lambda_j) x_{ij} & \\ \text{s.t.} & \sum_{i=1}^{m} x_{ij} \leq 1 \quad (\forall j) & (12) \\ & x_{ij} \leq y_i \quad (\forall i, j) & (13) \\ & y \in \{0,1\}^m, x \in \{0,1\}^{m \times n}. & \end{array}$$

We can now present the following result.

Proposition 1 *Let λ be given. For $i = 1, \ldots, m$, let $\bar{f}_i$ be the 'Lagrangian reduced cost' of facility i, according to the formula (9). Also, let T be the set of facility indices such that $\bar{f}_i \geq 0$. There exists an optimal solution to the relaxed problem such that $y_i = 0$ for all $i \in T$ (and therefore $x_{ij} = 0$ for all $i \in T$ and all j).*

Proof. Let (x^*, y^*) be a feasible solution to the relaxed problem, and suppose that there exists a facility $i \in T$ such that $y_i^* = 1$. We have:

$$f_i y_i^* + \sum_{j=1}^{n} (c_{ij} - \lambda_j) x_{ij}^* \geq f_i - \sum_{j=1}^{m} \max\left\{0, \lambda_j - c_{ij}\right\} = \bar{f}_i \geq 0.$$

Thus, we can obtain a solution to the relaxed problem that is no more costly than (x^*, y^*) by changing y_i from 1 to 0 and setting x_{ij} to 0 for all j. Repeating this procedure, as long as necessary, we obtain a solution such that $y_i = 0$ for all $i \in T$. □

In terms of the graph G_λ mentioned above, this means that, for any facility i with non-negative 'Lagrangian reduced cost', we can delete the corresponding node, along with all incident edges. (We have found that, in some cases, the removal of those nodes also causes some connected components of G_λ to break into two or more smaller components.)

3.3 Eliminating clients from the relaxed problem

Note that, after applying the first reduction procedure, there could be client nodes in G_λ with degree zero. Clearly, these can be eliminated from G_λ without affecting the solution of the relaxed problem. The following proposition enables us to eliminate client nodes of degree one as well.

Proposition 2 *Let λ be given, and let G_λ^- be the subgraph of G_λ that remains after the first reduction procedure. Let A be the set of clients for which the corresponding node in G_λ^- has degree one, and let B be the set of edges in G_λ^- that are incident on a node in A. There exists an optimal solution to the relaxed problem such that $x_{ij} = y_i$ for all $\{i,j\} \in B$.*

Proof. Let $\{i,j\}$ be an edge in B. If $y_i = 0$, then x_{ij} must be 0, due to constraints (13). If $y_i = 1$, then we would prefer to set x_{ij} to 1 rather than 0, since it would reduce the cost of the solution by $\lambda_j - c_{ij} > 0$. Now, given that the node for client j has degree 1 in G_λ^-, we can assume that $x_{i'j} = 0$ for all $i' \neq i$. Thus, we can safely set x_{ij} to 1 while satisfying the constraint (12) for the given client j. $\square$

A consequence of this proposition is that, for each $\{i,j\} \in B$, we can eliminate the variable x_{ij} from the relaxed problem, along with the constraints that it appears in, as long as we subtract $\lambda_j - c_{ij}$ from the cost of y_i. In terms of the graph G_λ^-, this means that we can delete the edge $\{i,j\}$ along with the client node j.

Note that our two reduction procedures do not improve the lower bound that we obtain from SLR for a given λ. Rather, they enable us to reduce the size of the relaxed problem, and thereby compute the lower bound more quickly.

3.4 A worked example

For clarity, we now show how our procedures work on a small example. Suppose that there are 7 facilities and 18 clients, located on a planar grid as shown in Figure 1. Here, the filled circles represent facilities and the small hollow circles represent clients. Each edge of the grid is supposed to have unit length. We also suppose that, for each facility i and client j, the assignment cost c_{ij} is equal to the rectilinear (a.k.a. Manhattan or taxicab) distance between the corresponding points. For example, $c_{11} = 1$, $c_{28} = 2$ and $c_{36} = 3$. Finally, we suppose that $f = (2, 2, 4, 2, 3, 1, 2)$.

The algorithm from [26] determines that the 'base level' is 2. That is, one can set λ_j to c_j^2 for all j, while still ensuring that $\bar{f}_i \geq 0$ for all i. To verify this, note that c_j^2 is equal to 1 for clients 2, 9, 10 and 13, equal to 3 for

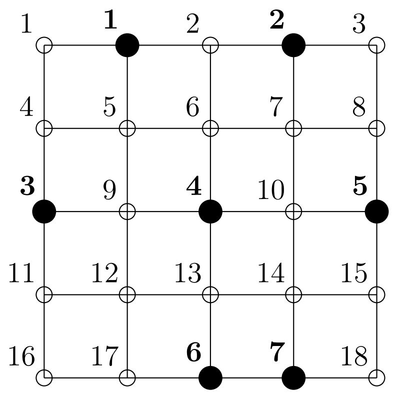

Figure 1: SPLP Instance.

client 11, and equal to 2 for the other 13 clients. One can then check that:

$$\begin{aligned}
\bar{f}_1 &= f_1 - \big(\lambda_1 - c_{1,1}\big) - \big(\lambda_5 - c_{1,5}\big) = 2 - 1 - 1 = 0\\
\bar{f}_2 &= f_2 - \big(\lambda_3 - c_{2,3}\big) - \big(\lambda_7 - c_{2,7}\big) = 2 - 1 - 1 = 0\\
\bar{f}_3 &= f_3 - \big(\lambda_4 - c_{3,4}\big) - \big(\lambda_{11} - c_{4,11}\big) = 4 - 1 - 2 = 1\\
\bar{f}_4 &= f_4 - \big(\lambda_6 - c_{4,6}\big) = 2 - 1 = 1\\
\bar{f}_5 &= f_5 - \big(\lambda_8 - c_{5,8}\big) - \big(\lambda_{15} - c_{5,15}\big) = 3 - 1 - 1 = 1\\
\bar{f}_6 &= f_6 - \big(\lambda_{17} - c_{6,17}\big) = 1 - 1 = 0\\
\bar{f}_7 &= f_7 - \big(\lambda_{14} - c_{7,14}\big) - \big(\lambda_{18} - c_{7,18}\big) = 2 - 1 - 1 = 0.
\end{aligned}$$

Thus, $\bar{f}_i \geq 0$ for all i as claimed. The optimal solution to the relaxed problem is trivial, since we open no facilities. The resulting lower bound is 33.

For completeness, we show the corresponding graph G_λ on the left of Figure 2. The edge from facility 1 to client 1 is present, for example, because $\lambda_1 - c_{11} = 2 - 1 = 1 > 0$. On the other hand, the edge from facility 1 to client 2 is not present, because $\lambda_2 - c_{12} = 1 - 1 = 0$.

Now, in step 4 of Algorithm 1, we add ϵ to all of the multipliers. We assume for simplicity that $\epsilon = 0.1$. One can check that this change in λ causes $\bar{f}$ to change from $(0, 0, 1, 1, 1, 0, 0)$ to $(-0.5, -0.5, 0.3, 0.1, 0.3, -0.7, -0.4)$. It also causes the number of edges in G_λ to increase substantially, from 12 to 44, as shown on the right of Figure 2. For example, the edge from facility 1 to client 2 is now present, because $\lambda_2 - c_{12}$ is now $1.1 - 1 = 0.1 > 0$.

We now apply our first reduction procedure. Since $\bar{f}_3$, $\bar{f}_4$ and $\bar{f}_5$ are non-negative, we can eliminate y_3, y_4 and y_5 from the relaxed problem, along with the associated x variables. The resulting subgraph of G_λ is shown on the left of Figure 3. (The three small crosses represent the facilities that have been eliminated from consideration.)

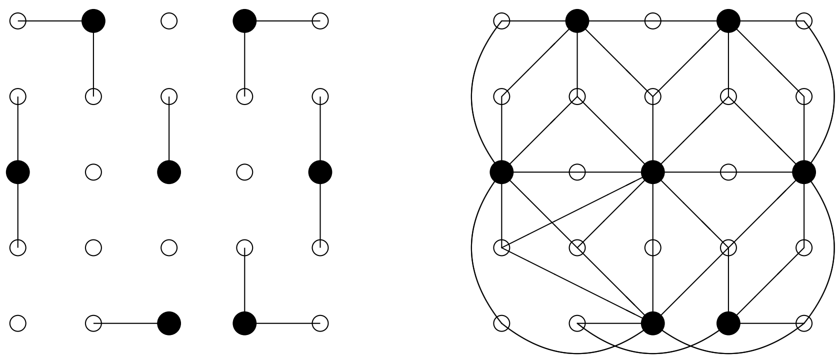

Figure 2: Graph G_λ at the base level (left) and after ϵ has been added (right).

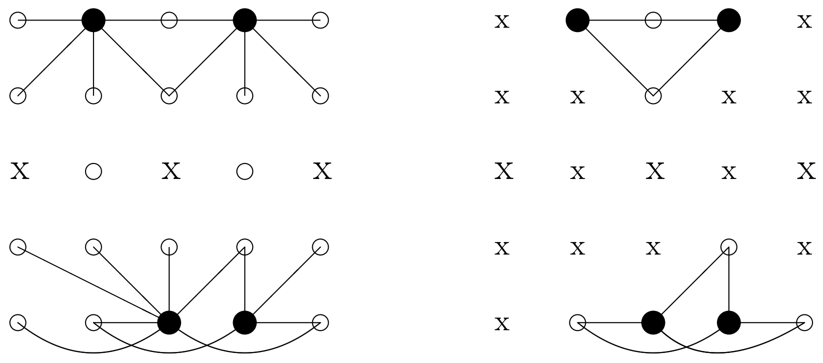

Figure 3: Subgraph G_λ after first new reduction procedure (left) and second new reduction procedure (right)

We see that the first reduction procedure has reduced the number of nodes from 25 to 22 and the number of edges from 44 to 21. Moreover, the graph now has two connected components, meaning that the relaxed problem can now be decomposed into two smaller subproblems.

Finally, we apply our second reduction procedure. Two of the client nodes have degree zero, so we can remove them from the graph. Another 11 client nodes have degree 1, so we can remove those nodes as well, along with the incident edges. The final reduced subgraph of G_λ is shown on the right of Figure 3. It has only 9 nodes and 10 edges.

For completeness, we mention that an optimal solution to the relaxed problem is to open facilities 2 and 6, yielding a lower bound of 33.6. Moreover, one can obtain a heuristic SPLP solution by keeping facilities 2 and 6 open, and then assigning each client to the nearest open facility. This yields an upper bound of 41.

4 Computational Experiments

In this section, we present computational results comparing our proposed approach with the BVA approach. For completeness, we also compare both approaches with the 'raw' `CPLEX` mixed-integer solver (with default settings).

Following Beltran *et al.* [6], we report results for 18 Barahona–Chudak (BC) instances [2] and 18 Koerkel–Ghosh (KG) instances [21, 16]. In addition, we report results for the 22 'M*' instances created by Kratica *et al.* [23]. All three sets of instances are available in the `UflLib` repository [18].

Following a suggestion of an anonymous referee, we created 55 additional large-scale instances. Details are given in Subsection 4.2.

All algorithms were implemented in C$^{\#}$, compiled with Visual Studio 2022, and run on a 2.4GHz Intel i5-1135G7 processor with 16GB of RAM under Windows 10. To solve all LPs and IPs, we used `CPLEX` v. 22.1, with default settings. A time limit of three hours was imposed for each algorithm across all instances.

In what follows, there is one subsection for each of the four sets of instances mentioned above. After that, in Subsection 4.5, we show how our new reduction procedures affect the graph G_λ, for a representative instance.

We remark that our implementation of the BVA method uses the first method described in Subsection 3.1 to determine the initial multiplier values. In our experience, this is slightly slower than the second method described in that subsection, but leads to slightly fewer iterations of the 'while' loop in Algorithm 1.

4.1 Barahona-Chudak (BC) instances

According to [2], the BC instances were generated as follows. In the unit square, n points are located at random. Each point is simultaneously a facility and a customer. The assignment costs c_{ij} are proportional to the Euclidean distances between the points. Facility opening costs are categorised as *small* ('s'), *medium* ('m'), or *large* ('l'), and are determined by the expression $\sqrt{n}/\ell$, where $\ell \in \{10, 100, 1000\}$. The values of n considered are $\{500, 1000, 1500, 2000, 2500, 3000\}$. This results in a total of $3 \times 6 = 18$ distinct instances. For interest, the optimal values for each instance can be found in the Appendix.

It turned out that 17 of the 18 BC instances could be solved to optimality by at least one of the three exact algorithms within the time limit. Table 1 shows the results for those 17 instances. The columns headed 'T1' show the time taken to compute the initial multipliers, in seconds. The columns headed 'T2' show the total time needed to solve the instance to proven optimality, again in seconds. If the algorithm did not succeed in solving the instance to optimality within the time limit, the entry is left blank. The columns headed 'Iter' show the number of iterations of the 'while' loop in

Algorithm 1. The columns headed '%E' show the percentage of the edges that remain in G_λ in the last iteration of the algorithm, after the reduction procedures have been applied. Finally, the last column shows the time taken by the `CPLEX`-IP solver to solve the instances to optimality, again in seconds.

We remark that the instance bc2500l could not be solved by any algorithm within the time limit. Moreover, `CPLEX` did not even manage to provide a lower bound.

		BVA				Ours				CPLEX-IP
Set	Name	T1	T2	Iter	%E	T1	T2	Iter	%E	T
BC	bc500s	3.33	10.20	5	0.36	0.01	2.67	5	0.26	12.39
	bc500m	3.80	8.36	15	1.70	0.02	1.02	11	1.00	11.32
	bc500l	17.46	103.62	81	7.08	0.04	17.22	55	2.90	16.46
	bc1000s	16.82	23.57	7	0.32	0.05	7.18	7	0.24	70.02
	bc1000m	20.28	61.16	35	1.34	0.10	18.28	26	0.67	83.55
	bc1000l	767.51	3,668.94	119	6.02	0.19	2,252.27	80	3.32	1,224.27
	bc1500s	42.01	49.72	7	0.29	0.12	4.50	7	0.21	405.21
	bc1500m	70.72	146.97	38	1.16	0.28	32.44	29	0.56	447.26
	bc1500l	1,860.74	8,764.55	138	5.25	1.02	5,357.03	90	2.54	3,200.17
	bc2000s	105.56	121.94	9	0.25	0.27	7.86	9	0.17	901.80
	bc2000m	312.07	630.13	42	1.07	0.57	181.32	29	0.50	1,410.95
	bc2000l	4,241.16	—	—	—	0.83	5,675.24	107	1.91	3,983.09
	bc2500s	109.69	125.47	10	0.23	0.39	9.38	9	0.16	613.61
	bc2500m	490.09	771.48	52	0.99	1.73	372.53	39	0.46	1,410.95
	bc3000s	178.30	203.19	12	0.22	1.13	20.76	11	0.15	990.89
	bc3000m	7,422.13	—	—	—	3.01	—	—	—	9,663.90
	bc3000l	7,466.17	—	—	—	4.33	6,081.26	90	1.33	5,968.17

Table 1: Results for the BC instances solved to optimality by at least one exact algorithm within the time limit.

A first observation is that the base-level approach for computing the initial multipliers is orders of magnitude faster than the LP-based approach. Moreover, when the base-level approach is used, the total number of iterations in the 'while' loop is reduced in most cases. This reduction occurs for 10 instances. This suggests that the base-level approach works very well, yielding good-quality multipliers quickly.

A second observation is that the BVA reduction procedure already eliminates a substantial portion of the edges in G_λ, even in the final iteration of the algorithm, when the graph is at its densest. Nevertheless, our reduction procedures eliminate a significant proportion of the remaining edges. The effect of this is that our version of the algorithm is consistently faster than the original. In fact, it solves two more instances within the time limit.

A third observation is that the computing time increases rapidly as the facility opening costs increase. A partial explanation is that the "base level" b, mentioned in Subsection 3.1, tends to increase as the opening costs increase. This in turn makes the graph G_λ more dense, which increases the burden on the IP solver. Moreover, in both SLR-based approaches, for each client j, the interval in which we search for the multiplier λ_j is $\left(c_j^{\ell(b)},\ \min_i\{c_{ij} + f_i\}\right]$. We found that, for the instances with large opening costs, the interval tended to be wider, which led to an increased number of iterations of the 'while' loop in Algorithm 1.

Compared to CPLEX-IP, our method is significantly faster on all instances with small or medium facility costs. However, CPLEX-IP tends to be faster for the instances with large facility costs. This is probably because, as mentioned above, those instances tend to need a high number of iterations. Although the relaxed subproblems are simpler than the original IP model, the repeated overhead of updating multipliers and building models increases the overall running time.

We were unable to solve two of the BC instances (bc2500l and bc3000m) using either SLR approach. As noted above, the `CPLEX`-IP solver was also unable to solve bc2500l and was close to the time limit for bc3000m.

4.2 Additional large-scale random planar instances

It is apparent from Table 1 that planar instances with small facility costs tend to be relatively easy for all three approaches. To explore this further, we created 11 larger instances of the same type, with m and n ranging from $5,000$ to $10,000$ in steps of 500. (The new instances will be made available at the Lancaster University Data Repository, under the heading 'Simple Plant Location Problem Random Planar Instances'.)[1]

It turned out that neither `CPLEX` nor the BVA algorithm were able to solve any of these larger instances within the time limit. Our algorithm, on the other hand, was able to solve all of the instances. The results for our algorithm are presented in Table 2. The columns have a similar interpretation to those in Table 1, except that the last two columns show the percentage of the edges that remained after the BVA reduction procedure, and our reduction procedures, respectively, in the last iteration of our algorithm.

As in the case of the BC instances, the number of iterations required by our algorithm was relatively small. This further reinforces that our base-level approach can generate good-quality multipliers. Moreover, our reduction procedures eliminated around half of the edges that remained after the BVA reduction procedure. (This seems to be the key to the success of our approach on these instances.)

Following an idea in Hansen *et al.* [17], we also generated a set of planar instances with 'varied' facility opening costs. These were generated in the same way as the previously mentioned instances, except that the f_i were integers drawn uniformly at random from the interval $[\sqrt{n}/1000, \sqrt{n}/10]$. Once more, the instance sizes vary from $5,000$ to $10,000$ in steps of 500.

[1] `http://www.research.lancs.ac.uk/portal/en/datasets/search.html`

Name	Opt	T1	T2	Iter	%E_1	%E_2
5000-s	541,341	3.36	93.82	31	0.17	0.08
5500-s	587,309	8.47	111.45	30	0.17	0.08
6000-s	630,075	7.88	174.19	38	0.16	0.07
6500-s	676,352	16.47	232.96	37	0.16	0.07
7000-s	721,928	18.63	222.36	32	0.15	0.07
7500-s	764,815	18.37	249.31	35	0.15	0.07
8000-s	807,556	19.77	376.89	43	0.15	0.07
8500-s	851,650	8.21	360.59	33	0.15	0.07
9000-s	894,522	10.97	553.9	45	0.14	0.07
9500-s	937,307	11.89	401.48	34	0.15	0.07
10000-s	977,953	10.4	1,114.13	46	0.14	0.06

Table 2: Results for large-scale planar instances with small facility costs.

As before, `CPLEX` was unable to solve any of these instances within the time limit. Moreover, the BVA method was unable to solve any of the instances apart from the smallest. On the other hand, our algorithm solved all of the instances. The results for our algorithm are shown in Table 3. Interestingly, the number of iterations was relatively high, but the new reduction procedures worked extremely well for these instances.

Name	Opt	T1	T2	Iter	%E_1	%E_2
5000-v	996,374	8.29	126.95	75	0.54	0.02
5500-v	1,076,968	8.86	150.99	75	0.51	0.01
6000-v	1,158,010	16.7	215.1	91	0.49	0.01
6500-v	1,230,124	20.37	200.53	71	0.47	0.01
7000-v	1,281,265	24.81	274.25	86	0.45	0.01
7500-v	1,338,209	27.87	315.86	86	0.42	0.01
8000-v	1,386,346	35.6	372.47	92	0.40	0.01
8500-v	1,483,334	12.98	374.53	80	0.40	0.01
9000-v	1,538,027	13.8	450.34	87	0.38	0.01
9500-v	1,606,715	19.03	678.87	106	0.38	0.01
10000-v	1,682,526	18.32	555.74	88	0.38	0.01

Table 3: Results for large-scale planar instances with varied facility costs.

4.3 Koerkel-Ghosh instances

For the KG instances [21, 16], the c_{ij} are integers drawn uniformly at random from $[1000, 2000]$. The f_i are integers drawn uniformly at random from $[100, 200]$ in class A, from $[1000, 2000]$ in class B, and from $[10000, 20000]$ in class C. As in the case of the BC instances, we have $m = n$. There are 'symmetric' instances, in which $c_{ij} = c_{ji}$, and 'asymmetric' ones, in which

c_{ij} and c_{ji} are generated independently. As in [6], we have selected one symmetric and one asymmetric instance from each class and for each value of $m = n$ in $\{250, 500, 750\}$, making $2 \times 3 \times 3 = 18$ instances in total.

At the time of writing, only 8 of the selected KG instances have been solved to proven optimality. The optimal values for those 8, and the best known lower and upper bounds for the remaining 10, can be found in the Appendix.

It turned out that the BVA algorithm was not able to solve any of the KG instances to proven optimality within the time limit. Moreover, both our algorithm and `CPLEX` were able to solve only two of the instances. The results for those two instances can be found in Table 4. It can be seen that our reduction procedures still eliminated a significant proportion of the edges, but not as much as for the planar instances. Moreover, `CPLEX` is faster than our method on one instance.

	Ours				CPLEX-IP
	T1	T2	Iter	%E	T
gs250a-1	0.01	3,230.82	45	3.43	2,609.53
ga250a-1	0.01	2,489.02	33	3.52	3,097.73

Table 4: Results for the two KG instances that CPLEX-IP and our algorithm could solve to optimality within the time limit.

For the remaining 16 KG instances, we followed a different approach. For each instance and each SLR-based method, we recorded the lower bound at the end of the time limit. Moreover, we constructed a feasible SPLP solution as follows: we took the solution to the last relaxation, and completed it by assigning each unassigned client to the open facility with the lowest assignment cost. In this way, we obtained an upper bound as well as a lower bound.

The results for the 16 KG instances are shown in Table 5. For each SLR-based approach, we report (i) the percentage difference between the lower bound produced by that approach and the best known lower bound ('Δ_{LB} (%)'), (ii) the gap between the lower and upper bounds produced by the same approach ('Gap_{UB} (%)'), and (iii) the number of iterations ('Iter'). The Δ_{LB} (%) value is computed as $\Delta_{\text{LB}} = \frac{\text{LB}-\text{LB}^*}{\text{LB}^*} \times 100\%$, where LB^* is the best known lower bound, and LB is the lower bound produced by the evaluated approach. Positive values indicate that the approach achieved a stronger lower bound than the reference, while negative values indicate a weaker lower bound. The Gap_{UB} is calculated as $\text{Gap}_{\text{UB}} = \frac{\text{UB}-\text{LB}}{\text{LB}} \times 100\%$, where UB and LB are the upper and lower bounds obtained by the same approach.

We see that our approach produces stronger average lower bounds than the BVA approach. Moreover, the gap between the lower and upper bounds produced by our approach is smaller than that produced by BVA in almost all cases. The only exception is the instance 'ga500a-1', for which the BVA bound is slightly larger than ours. (We remark that, by allowing our algorithm an additional 6 minutes on that instance, we obtained a stronger lower bound than the one found by the BVA algorithm.) It is also apparent that our algorithm typically took fewer iterations than the BVA algorithm.

	BVA			Ours		
Name	Δ_{LB} (%)	Gap_{UB} (%)	Iter	Δ_{LB} (%)	Gap_{UB} (%)	Iter
gs250b-1	0.62	0.61	52	0.65	0.52	41
gs250c-1	-0.29	1.25	209	-0.24	0.76	164
ga250b-1	0.48	0.57	59	0.52	0.62	50
ga250c-1	-0.44	0.90	191	-0.37	1.58	145
gs500a-1	-0.06	0.55	12	-0.03	0.51	9
gs500b-1	-0.21	2.83	34	-0.09	1.70	20
gs500c-1	1.22	5.46	116	1.51	2.54	59
ga500a-1	-0.05	0.70	12	-0.06	0.60	7
ga500b-1	-0.20	2.68	35	-0.07	1.66	19
ga500c-1	1.32	5.78	119	1.59	2.47	59
gs750a-1	-0.08	0.95	11	-0.08	0.93	7
gs750b-1	-0.19	3.48	33	-0.09	3.23	14
gs750c-1	-0.87	7.96	107	-0.43	5.08	43
ga750a-1	-0.08	0.83	11	-0.05	0.94	7
ga750b-1	-0.28	4.73	32	-0.13	2.05	13
ga750c-1	-0.75	8.73	108	-0.37	5.12	39
Averages	0.01	3.00		0.14	1.89	

Table 5: Results for the 16 KG instances that were not solved to optimality by any algorithm within the time limit.

4.4 Kratica *et al.* instances

The M* instances [23] have random facility and assignment costs, but there is a negative correlation between the facility costs and the assignment costs. There are 22 instances in total; five each for $m = n$ in $\{100, 200, 300, 500\}$, one instance with $m = n = 1000$, and another with $m = n = 2000$. The optimal value for each instance, taken from [6, 14], can be found in the Appendix.

Table 6 shows results for instances that can be solved to optimality by at least one of the three exact methods within the time limit. Consistent with the other results, the base-level approach for computing the initial multipliers is substantially faster than the LP-based approach. Specifically, achieving average speedups of over 1,000 times for the M* instances reported in the table. Interestingly, for all M* instances, the number of iterations required by our method is much lower than for the BVA method.

The BVA reduction procedure eliminates a substantial portion of the edges in G_λ in the final iteration of the algorithm, although not to the same degree as for the planar instances, and our reduction procedures still eliminate a significant proportion of those remaining edges. The BVA algorithm removes an average of 61.18% of the edges, whereas our approach achieves an average reduction of 73.53%.

For the M* instances, our method performs significantly slower than CPLEX-IP. This observation is consistent with the results for BC instances with large facility costs and can be attributed to the high number of iterations required by our approach. Additionally, CPLEX solved three more M* instances than both the SLR approaches within the time limit, despite the amount of time needed to find the multipliers being relatively low in both cases.

	BVA				Ours				CPLEX-IP
Name	T1	T2	Iter	%E	T1	T2	Iter	%E	T
MO1	0.24	69.50	95	39.41	0.00	38.14	71	28.40	4.88
MO2	0.20	64.64	100	39.50	0.00	43.33	81	29.85	3.56
MO3	0.25	61.51	83	41.19	0.00	31.82	57	32.27	4.14
MO4	0.24	25.49	72	36.27	0.00	5.71	50	14.77	2.76
MO5	0.24	75.38	106	40.33	0.00	37.27	83	31.24	2.74
MP1	6.89	675.86	171	36.17	0.01	278.39	130	17.65	41.83
MP2	10.71	1,530.81	205	40.03	0.01	1,055.05	154	37.01	48.73
MP3	7.98	410.87	162	36.34	0.01	151.48	119	12.28	22.60
MP4	8.62	1,851.50	226	40.73	0.01	1,154.96	176	35.26	65.09
MP5	7.46	3,655.97	304	43.65	0.01	3,217.05	257	43.65	76.89
MQ1	23.80	3,333.54	264	37.79	0.02	1,664.61	189	22.92	198.04
MQ2	19.61	3,249.45	250	34.84	0.02	1,110.73	185	17.56	322.28
MQ3	26.09	3,465.16	287	39.41	0.02	1,822.60	214	24.38	129.99
MQ4	18.96	3,635.86	284	39.34	0.02	1,691.91	210	21.81	216.98
MQ5	20.38	10,559.02	343	37.36	0.02	6,698.20	278	27.97	622.87
MR1	82.91	—	—	—	0.05	—	—	—	5,590.92
MR2	76.98	—	—	—	0.05	—	—	—	4,764.60
MR5	102.69	—	—	—	0.05	—	—	—	9,534.60

Table 6: Results for the M* instances solved to optimality by at least one exact algorithm within the time limit.

Table 7 shows the results for the four M* instances that none of the exact methods were able to solve to optimality within the time limit. We see that, once more, our approach produces stronger average lower bounds than BVA. The average $\Delta_{LB}(\%)$ for the BVA approach was -1.20 compared with 1.88 for our approach. Notably, for the largest instances, the BVA approach has a weaker lower bound than the reference while our approach achieves a stronger bound. The gap between the lower and upper bounds produced by our approach is smaller than that produced by BVA in all cases. As noted previously, our algorithm ran for fewer iterations than the BVA algorithm.

	BVA			Ours		
Name	Δ_{LB} (%)	Gap_{UB} (%)	Iter	Δ_{LB} (%)	Gap_{UB} (%)	Iter
MR3	3.26	8.10	231	3.97	1.92	156
MR4	1.91	3.54	236	2.52	2.63	168
MS1	2.01	10.29	331	4.66	7.15	233
MT1	-12.77	117.82	361	3.32	23.14	232
Averages	-1.20	21.09		1.88	5.71	

Table 7: Results for M* instances that were not solved to optimality by either exact algorithm within the time limit.

4.5 Evolution of the graph for one instance

Finally, to give the reader more insight into our reduction procedures, we show how the procedures affect the evolution of the graph G_λ as the algorithm proceeds. For brevity, we use just one instance: the BC instance with $m = n = 1500$ and $\ell = 1000$.

The results are given in Table 8. Each row corresponds to one iteration of the 'while' loop in Algorithm 1 and ϵ is set to 0.1. (The row for 'iteration 0' corresponds to the initial multipliers that were obtained with the base-level method.) The first column shows the iteration number and the second column shows the lower bound. The column headed 'E0' shows the number of edges that remain after the reduction procedure from [6] has been applied. The columns headed 'E1' and 'E2' show the number that still remain after our first and second reduction procedures have been applied, respectively. The column headed '$\tilde{m}$' shows the number of facility nodes remaining after our first procedure, and the column headed '$\tilde{n}$' shows the number of client nodes remaining after our second procedure. The column headed #CCs gives the number of connected components after all reduction steps have been taken for each iteration.

Iter.	LB	E0	E1	E2	$\tilde{m}$	$\tilde{n}$	#CCs
1	150	1,564	0	0	0	0	0
2	207,985	3,249	16	12	10	6	0
3	302,980	4,938	708	573	234	259	7
4	332,873	6,230	3,901	3,673	936	1,208	109
5	334,675	6,401	4,688	4,555	1,072	1,367	92
6	334,925	6,425	4,736	4,611	1,079	1,375	51
7	334,962	6,431	4,768	4,646	1,084	1,378	48

Table 8: Evolution of G_λ for one specific BC instance.

We see that all three reduction procedures substantially reduce the number of edges. (Note that, before the reductions, the graph contains $1500^2 = 2,250,000$ edges.) The new procedures also reduce the number of nodes substantially. Moreover, the new procedures are especially useful in the earlier iterations of the algorithm (when most of the facilities still have a non-negative reduced cost).

5 Conclusions

The SLR approach to the SPLP, as presented in [6], was already a very promising approach. Our enhancements to the approach included a faster procedure for initialising the multipliers, and two procedures for reducing the number of variables in the relaxed problem. The computational results demonstrate that the proposed method consistently outperforms the original algorithm by finding good initial multipliers faster, reducing iterations, and achieving better lower bounds across most instances. However, for the more challenging KG instances, while it generally produces stronger bounds and fewer iterations, it struggles to solve many instances within the time limit.

Compared with the CPLEX-IP solver, our method performs strongly on the random planar instances, while the results are more mixed for the BC instances. For most of the M* instances, our approach yields inferior results, but it provides improved lower bounds for the four largest instances in that set. On the more challenging KG instances, our method outperforms CPLEX-IP on 12 out of 18 instances, but still performs worse than the Benders decomposition method in [11].

An interesting question for further research is whether one could obtain even better results by combining our enhanced SLR approach with the 'aggressive reduction' procedures presented in [27] or the 'aggressive multiplier adjustment' procedure presented in [20]. Potentially, this could enable even larger SPLP instances to be solved. Another promising direction is the application of SLR to other combinatorial optimisation problems, besides the ones considered in [4, 5, 6, 8, 31]. A further potential research direction is, for large-scale instances, to strengthen the formulations of subproblems in order to further improve the LBs. For example, one could consider using the tighter formulations such as the one proposed by Elloumi [12].

Acknowledgement: The first author gratefully acknowledges financial support from the EPSRC, through the STOR-i Centre for Doctoral Training, under grant STOR-i 3 - EP/S022252/1.

A Best Known Bounds for Benchmark Instances

Table 9 shows the optimal values for the Barahona-Chudak (BC) instances, taken from [6, 14] and also verified by ourselves.

Name	Opt	Name	Opt
bc500s	99,169	bc2000s	437,686
bc500m	326,790	bc2000m	1,122,748
bc500l	798,577	bc2000l	2,558,118
bc1000s	220,560	bc2500s	534,405
bc1000m	607,878	bc2500m	1,347,516
bc1000l	1,434,154	bc2500l	3,099,907
bc1500s	334,962	bc3000s	643,463
bc1500m	866,454	bc3000m	1,602,154
bc1500l	2,000,801	bc3000l	3,570,766

Table 9: Optimal values for the BC instances.

The Koerkel-Ghosh (KG) instances, on the other hand, have not all been solved to proven optimality. Table 10 shows the optimal values (in bold) for the 8 instances which have been solved, and ranges on the optimal values for the remaining 10 instances. For the latter instances, the lower bounds were taken from [11] and the upper bounds were taken from [6, 14, 29].

Name	Opt	Name	Opt
gs250a-1	**257,964**	ga250a-1	**257,957**
gs250b-1	**276,761**	ga250b-1	**276,296**
gs250c-1	**332,935**	ga250c-1	**334,135**
gs500a-1	[510,803, 511,187]	ga500a-1	[510,911, 511,383]
gs500b-1	[535,349, 537,931]	ga500b-1	[535,814, 538,060]
gs500c-1	**620,041**	ga500c-1	**621,360**
gs750a-1	[762,856, 763,671]	ga750a-1	[762,723, 763,520]
gs750b-1	[792,289, 797,026]	ga750b-1	[792,148, 796,454]
gs750c-1	[888,979, 900,363]	ga750c-1	[888,930, 902,026]

Table 10: Optimal values (in bold) for 8 solved KG instances, and ranges for the 10 unsolved instances.

Finally, Table 11 shows the optimal values for the M* instances, taken from [6, 14] and also verified by ourselves.

Name	$m = n$	Opt	Name	$m = n$	Opt
MO1		1,305.95	MQ1		4,091.01
MO2		1,432.36	MQ2		4,028.33
MO3	100	1,516.77	MQ3	300	4,275.43
MO4		1,442.24	MQ4		4,235.15
MO5		1,408.77	MQ5		4,080.74
MP1		2,686.48	MR1		2,608.15
MP2		2,904.86	MR2		2,654.73
MP3	200	2,623.71	MR3	500	2,788.25
MP4		2,938.75	MR4		2,756.04
MP5		2,932.33	MR5		2,505.05
MS1	1000	5,283.76	MT1	2000	10,069.80

Table 11: Optimal values for the M* instances.

References

[1] M.L. Balinski. Integer programming: methods, uses, computations. *Manag. Sci.*, 12:253–313, 1965.

[2] F. Barahona and F.A. Chudak. Near-optimal solutions to large-scale facility location problems. *Discr. Optim.*, 2:35–50, 2005.

[3] J.E. Beasley. Lagrangian heuristics for location problems. *Eur. J. Oper. Res.*, 65:383–399, 1993.

[4] I. Belik and K. Jörnsten. A new semi-Lagrangean relaxation for the k-cardinality assignment problem. *J. Infor. Optim. Sci.*, 37:75–100, 2016.

[5] C. Beltrán-Royo, C. Tadonki, and J.-P. Vial. Solving the p-median problem with a semi-Lagrangian relaxation. *Comput. Optim. Appl.*, 35:239–260, 2006.

[6] C. Beltrán-Royo, J.-P. Vial, and A. Alonso-Ayuso. Semi-Lagrangian relaxation applied to the uncapacitated facility location problem. *Comput. Optim. Appl.*, 51:387–409, 2012.

[7] O. Bilde and J. Krarup. Sharp lower bounds and efficient algorithms for the simple plant location problem. *Ann. Discr. Math.*, 1:79–97, 1977.

[8] X. Cabezas and S. García. A semi-Lagrangian relaxation heuristic algorithm for the simple plant location problem with order. *J. Oper. Res. Soc.*, 74:2391–2402, 2023.

[9] X. Cabezas, S. García, C. Martin-Barreiro, E. Delgado, and V. Leiva. A two-stage location problem with order solved using a Lagrangian al-

gorithm and stochastic programming for a potential use in COVID-19 vaccination based on sensor-related data. *Sensors*, 21(16):5352, 2021.

[10] G. Cornuéjols, G.L. Nemhauser, and L.A Wolsey. The uncapacitated facility location problem. In P.B. Mirchandani and R.L. Francis, editors, *Discrete Location Theory*, pages 1–54. Wiley, New York, 1990.

[11] C. Durán-Mateluna, Z. Alés, and S. Elloumi. An efficient Benders decomposition for the p-median problem. *Eur. J. Oper. Res.*, 308:84–96, 2023.

[12] S. Elloumi. A tighter formulation of the p-median problem. *J. Comb. Optim.*, 19(1):69–83, 2010.

[13] D. Erlenkotter. A dual-based procedure for uncapacitated facility location. *Oper. Res.*, 26:992–1009, 1978.

[14] M. Fischetti, I. Ljubić, and M. Sinnl. Redesigning Benders decomposition for large-scale facility location. *Manag. Sci.*, 63:2146–2162, 2017.

[15] R.D. Galvão and L.A. Raggi. A method for solving to optimality uncapacitated location problems. *Ann. Oper. Res.*, 18:225–244, 1989.

[16] D. Ghosh. Neighborhood search heuristics for the uncapacitated facility location problem. *Eur. J. Oper. Res.*, 150:150–162, 2003.

[17] P. Hansen, J. Brimberg, D. Urošević, and N. Mladenović. Primal-dual variable neighborhood search for the simple plant-location problem. *INFORMS J. Comput.*, 19:552–564, 2007.

[18] M. Hoefer. UflLib: a library of instances of the uncapacitated facility location problem, 2006. Department of Algorithms and Complexity, Max Plank Institute for Informatics, Saarbrücken.

[19] J. Janáček and L. Buzna. An acceleration of Erlenkotter–Körkel's algorithms for the uncapacitated facility location problem. *Ann. Oper. Res.*, 164:97–109, 2008.

[20] K. Jörnsten and A. Klose. An improved Lagrangian relaxation and dual ascent approach to facility location problems. *Comput. Manag. Sci.*, 13:317–348, 2016.

[21] M. Körkel. On the exact solution of large-scale simple plant location problems. *Eur. J. Oper. Res.*, 39:157–173, 1989.

[22] J. Krarup and P.M. Pruzan. The simple plant location problem: survey and synthesis. *Eur. J. Oper. Res.*, 12:36–81, 1983.

[23] J. Kratica, D. Tošic, V. Filipović, and I. Ljubić. Solving the simple plant location problem by genetic algorithm. *RAIRO Oper. Res.*, 35:127–142, 2001.

[24] M. Labbé and F. Louveaux. Location problems. In M. Dell'Amico, F. Maffoli, and S. Martello, editors, *Annotated Bibliographies in Combinatorial Optimization*, pages 261–281. Wiley, Chichester, 1997.

[25] G. Laporte, S. Nickel, and F. Saldanha da Gama, editors. *Location Science*. Springer, Cham, 2019.

[26] A.N. Letchford and S.J. Miller. Fast bounding procedures for large instances of the simple plant location problem. *Comput. Oper. Res.*, 39:985–990, 2012.

[27] A.N. Letchford and S.J. Miller. An aggressive reduction scheme for the simple plant location problem. *Eur. J. Oper. Res.*, 234:674–682, 2014.

[28] E. Monabbati. An application of a Lagrangian-type relaxation for the uncapacitated facility location problem. *Jpn. J. Ind. Appl. Math.*, 31:483–499, 2014.

[29] M. Posta, J.A. Ferland, and P. Michelon. An exact cooperative method for the uncapacitated facility location problem. *Math. Program. Comput.*, 6:199–231, 2014.

[30] V. Verter. Uncapacitated and capacitated facility location problems. In H.A. Eiselt and V. Marianov, editors, *Foundations of Location Analysis*, pages 25–28. Springer, Berlin, 2011.

[31] H. Zhang, C. Beltrán-Royo, B. Wang, L. Ma, and Z. Zhang. Solution to the quadratic assignment problem using semi-Lagrangian relaxation. *J. Syst. Eng. Electron.*, 27:1063–1072, 2016.

[32] H. Zhang, C. Beltrán-Royo, B. Wang, and Z. Zhang. Two-phase semi-Lagrangian relaxation for solving the uncapacitated distribution centers location problem for B2C E-commerce. *Comput. Optim. Appl.*, 72:827–848, 2019.