A Separation Algorithm for the Simple Plant Location Problem

Laura Galli* Adam N. Letchford†

To appear in *Operations Research Letters*

Abstract

The *Simple Plant Location Problem* (SPLP) is a well-known $\mathcal{NP}$-hard optimisation problem with applications in logistics. Although many families of facet-defining inequalities are known for the associated polyhedron, very little work has been done on separation algorithms. We present the first ever polynomial-time separation algorithm for the SPLP that separates exactly over an exponentially large family of facet-defining inequalities. We also present some promising computational results.

Keywords: facility location; combinatorial optimisation; branch-and-cut

1 Introduction

The *Simple Plant Location Problem* (SPLP), also known as the *Uncapacitated Facility Location Problem*, is a well-known optimisation problem arising in logistics applications (see, e.g., [14, 23, 25, 30]). We are given a set I of facilities and a set J of clients. The cost of opening facility $i \in I$ is denoted by f_i, and the cost of assigning client $j \in J$ to facility $i \in I$ is c_{ij}. The task is to decide which facilities to open, and to assign each client to an open facility, at minimum cost.

The SPLP is an $\mathcal{NP}$-hard combinatorial optimisation problem [3, 23]. The standard integer programming formulation of the SPLP is due to Balinski [3]. Let m denote $|I|$ and n denote $|J|$. For each $i \in I$, define a binary variable y_i, taking the value 1 if and only if facility i is opened. For each $i \in I$ and $j \in J$, define a binary variable x_{ij}, taking the value 1 if and only

*Dipartimento di Informatica, Università di Pisa, Largo B. Pontecorvo 3, 56127 Pisa, Italy. E-mail: Laura.Galli@unipi.it

†Department of Management Science, Lancaster University Management School, Lancaster LA1 4YX, United Kingdom. E-mail: A.N.Letchford@lancaster.ac.uk

if client j is assigned to facility i. The formulation is then:

$$\begin{aligned}
\min \quad & \textstyle\sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{i\in I} f_i y_i && && (1)\\
\text{s.t.} \quad & \textstyle\sum_{i\in I} x_{ij} = 1 && (j \in J) && (2)\\
& x_{ij} \le y_i && (i \in I, j \in J) && (3)\\
& (x, y) \in \{0,1\}^{mn+m}. && && (4)
\end{aligned}$$

We call (2) *assignment constraints* and (3) *variable upper bounds* (VUBs).

Many families of valid and facet-defining inequalities have been derived for the above formulation [5, 10, 11, 13, 15, 17, 19]. On the other hand, very little work has been done on separation algorithms. Indeed, until the present paper, exact polynomial-time separation algorithms were known only for one family of inequalities, the so-called *odd cycle* inequalities [6, 7, 8]. Moreover, the odd cycle inequalities do not define facets in general [11, 15].

In this paper, we focus on some facet-defining inequalities due to Cho *et al.* [11], which we call *3-client* inequalities. Our main contribution is to describe an exact separation algorithm for the 3-client inequalities. To our knowledge, our algorithm is the first ever separation algorithm for the SPLP that (a) runs in polynomial time, and (b) separates exactly over an exponentially large family of facet-defining inequalities.

The paper is structured as follows. Section 2 is a brief literature review. Section 3 presents the separation algorithm. (A faster separation algorithm is also presented for a less general family of inequalities.) In Section 4, we present some computational results and some concluding remarks.

Throughout the paper, we assume that the reader is familiar with the polyhedral approach to combinatorial optimisation (see [9, 12, 22]). We let $P(m, n)$ denote the convex hull of all pairs $(x, y) \in \mathbb{R}^{mn+m}$ that satisfy (2)–(4). We sometimes write $x(i, j)$ for x_{ij} and $y(i)$ for y_i. Moreover, sometimes we write $x(S : T)$ for $\sum_{i\in S}\sum_{j\in T} x_{ij}$ and $y(S)$ for $\sum_{i\in S} y_i$. Finally, when we say "trivial bounds", we mean the constraints $x_{ij} \ge 0$ for all $i \in I$ and $j \in J$ and the constraints $y_i \le 1$ for all $i \in I$.

2 Literature Review

Since the literature on the SPLP is vast, we focus here on papers of direct relevance, and refer the reader to the surveys [14, 23, 25, 30] for more details. We cover valid inequalities in Subsection 2.1 and separation algorithms in Subsection 2.2.

2.1 Some valid inequalities

Cornuéjols and Thizy [15] showed that the assignment constraints give a complete and non-redundant description of the affine hull of $P(m, n)$. They also showed that the VUBs and trivial bounds define facets. Cho *et al.* [10]

showed that $P(m,n)$ is completely described by the assignment constraints, VUBs and trivial bounds when at least one of m and n is less than 3. Otherwise, more inequalities are needed.

Let $p \geq 3$ be an odd integer. Let $s_1, \ldots, s_p$ be distinct location indices, let $t_1, \ldots, t_p$ be distinct client indices, and take indices modulo p, so that s_{p+1} is identified with s_1. The *odd cycle* inequality

$$\sum_{i=1}^{p} \big(x(s_i,t_i) + x(s_i,t_{i+1})\big) \;\leq\; \sum_{i=1}^{p} y(s_i) + \lfloor p/2 \rfloor \tag{5}$$

is valid [10, 15]. Odd cycle inequalities define facets if and only if $p = 3$ [11, 15]. For larger p, they must be strengthened ("lifted") to make them facet-defining [11, 15].

Following [17], we will call the odd cycle inequalities with $p = 3$ *3-cycle* inequalities. Cho *et al.* [10] showed that the 3-cycle inequalities, together with the assignment constraints, VUBs and trivial bounds, give a complete description of $P(m,n)$ when $m = 3$.

Cho *et al.* [11] presented an interesting generalisation of the 3-cycle inequalities. Let j_1, j_2 and j_3 be three client indices, and let S_{12}, S_{13}, S_{23} and S_{123} be disjoint subsets of I. (One of the four subsets is permitted to be empty.) Then the following inequality is valid and facet-defining:

$$\begin{aligned} 2\,x\big(S_{12} : \{j_1,j_2\}\big) \;+\; 2\,x\big(S_{13} : \{j_1,j_3\}\big) \;+\; 2\,x\big(S_{23} : \{j_2,j_3\}\big) \;+ \\ x\big(S_{123} : \{j_1,j_2,j_3\}\big) \;\leq\; 2 \;+\; 2\,y\big(S_{12} \cup S_{13} \cup S_{23}\big) \;+\; y\big(S_{123}\big). \end{aligned} \tag{6}$$

We will call (6) *3-client* inequalities. Note that the 3-client inequalities with $\big|S_{12}\big| = \big|S_{13}\big| = \big|S_{23}\big| = 1$ and $S_{123} = \emptyset$ are equivalent to the 3-cycle inequalities. Cho *et al.* [10] showed that the 3-client inequalities, together with the assignment constraints, VUBs and trivial bounds, give a complete description of $P(m,n)$ when $n = 3$.

Several additional families of inequalities are presented in [1, 5, 10, 11, 13, 15, 17, 19]. We omit details for brevity.

2.2 Separation algorithms

To our knowledge, before the present paper, exact polynomial-time separation algorithms were known only for the odd cycle inequalities (5).

Caprara & Fischetti [6] showed that odd-cycle separation can be reduced to a series of $O(mn)$ shortest (s,t)-path problems in a graph with $O(mn)$ nodes and $O\big(m^2 n\big)$ edges. Although this algorithm runs in polynomial time, it is too slow to be useful in practice.

A faster odd-cycle separation algorithm, running in $O\big(m^2(m+n)\big)$ time, was given in [7]. Although this is still rather slow, the algorithm actually returns m odd cycle inequalities, each of which can be checked for violation. Computational results obtained with this algorithm are given in [8].

Unfortunately, the results in [8] indicate that odd cycle inequalities typically close only a small proportion of the integrality gap. This may be due to the fact, mentioned above, that the inequalities do not define facets in general.

The only other separation algorithm we are aware of is due to Aardal [1]. In Subsection 5.3 of her paper, she defines a family of inequalities that generalises the odd cycle inequalities, and devises a heuristic separation algorithm for them. The inequalities are not facet-defining in general. As far as we can tell, the separation heuristic runs in $O\big(m^2n^2\big)$ time, but it can yield up to m violated inequalities in a single call.

To close this section, we mention that, at present, the leading exact algorithms for the SPLP are the ones in [16, 26, 28]. Strangely, none of them exploit the above-mentioned results on polyhedra and separation. A possible explanation is that, for many instances, the continuous relaxation of the Balinski formulation already gives a reasonably tight lower bound [2, 27].

3 Separation Algorithms

We now turn our attention to separation. In Subsection 3.1, we present an exact separation algorithm for the 3-client inequalities (6), which runs in $O\big(mn^3\big)$ time. In Subsection 3.2, we discuss ways to speed up the algorithm in practice. Then, in Subsection 3.3, we present an exact separation algorithm for the (less general) 3-cycle inequalities, which runs in $O\big(m^2n\big)$ time.

Throughout this section, we let (x^*, y^*) denote the fractional point to be separated. We assume w.l.o.g. that (x^*, y^*) satisfies the assignment constraints (2), the VUBs (3) and the trivial bounds.

3.1 Separation for 3-client inequalities

myblueBefore presenting our separation algorithm for 3-client inequalities, we will need the following lemma.

Lemma 1 *If two or more of the sets S_{12}, S_{13}, S_{23} and S_{123} are empty, then the 3-client inequality (6) remains valid, but it is implied by the assignment constraints (2) and VUBs (3).*

Proof. If S_{13} and S_{23} are empty, then the 3-client inequality reduces to

$$2\,x\big(S_{12} : \{j_1, j_2\}\big) \;+\; x\big(S_{123} : \{j_1, j_2, j_3\}\big) \;\leq\; 2 \;+\; 2\,y\big(S_{12}\big) \;+\; y\big(S_{123}\big).$$

This is implied by (a) the assignment constraints on j_1 and j_2, (b) the VUBs on S_{12} and $\{j_1, j_2\}$, and (c) the VUBs on S_{123} and j_3. A similar argument

applies if S_{12} and S_{13} are empty, or S_{12} and S_{23} are empty. If S_{23} and S_{123} are empty, then the 3-client inequality reduces to

$$2\,x\big(S_{12} : \{j_1, j_2\}\big) \;+\; 2\,x\big(S_{13} : \{j_1, j_3\}\big) \;\leq\; 2 \;+\; 2\,y\big(S_{12} \cup S_{13}\big).$$

This is implied by (a) the assignment constraint on j_1 (taken twice), (b) the VUBs on S_{12} and j_2, and (c) the VUBs on S_{13} and j_3. □

We are now ready to present our separation algorithm for 3-client inequalities.

Proposition 1 *The separation problem for the 3-client inequalities (6) can be solved exactly in $O\big(mn^3\big)$ time.*

Proof. There are $O\big(n^3\big)$ possible ways to select the client triple $\{j_1, j_2, j_3\}$. For each such triple, we can find the facility sets S_{12}, S_{13}, S_{23} and S_{123} that minimise the slack of the inequality (or, equivalently, maximise the violation) as follows. For each $i \in M$, compute the following four quantities:

$$\begin{gathered} 2\left(x^*(i, j_1) + x^*(i, j_2) - y_i^*\right) \\ 2\left(x^*(i, j_1) + x^*(i, j_3) - y_i^*\right) \\ 2\left(x^*(i, j_2) + x^*(i, j_3) - y_i^*\right) \\ x^*(i, j_1) + x^*(i, j_2) + x^*(i, j_3) - y_i^*. \end{gathered}$$

If at least one of these quantities is positive, then place i in S_{12}, S_{13}, S_{23} or S_{123}, respectively, according to which of the quantities is the largest. Otherwise, do not place i in any of those sets.

Note that, for a given client triple and a given i, one can compute the four quantities in constant time. Thus, for a given triple, one can compute S_{12}, S_{13}, S_{23} and S_{123}, and check the resulting inequality for violation, in $O(m)$ time.

One small complication is that, for a given client triple, more than one of the sets S_{12}, S_{13}, S_{23} and S_{123} may turn out to be empty. If that happens, then, by Lemma 1, no 3-client inequality is violated for that triple. □

3.2 Exploiting sparsity

Since a running time of $O\big(mn^3\big)$ is rather high, we now show how to speed up the algorithm, by exploiting the sparsity of the LP solution. We will need the following three lemmas.

Lemma 2 *If a 3-client inequality is violated, then $y_i^* < 1$ for all $i \in S_{12} \cup S_{13} \cup S_{23} \cup S_{123}$.*

Proof. Suppose that $y_i^* = 1$ for some $i \in S_{12}$. This implies that y^* satisfies $y(S_{12}) \geq 1$ or, equivalently, $-2y(S_{12}) \leq -2$. Add to this (i) two times the assignment constraints on clients j_1 and j_2, (ii) two times the VUBs for the client j_3 and the facilities in S_{13} and S_{23}, and (iii) the VUBs for the client j_3 and the facilities in S_{123}. The resulting inequality dominates the 3-client inequality.

The cases in which i belongs to S_{13} or S_{23} are similar.

Now suppose that $y_i^* = 1$ for some $i \in S_{123}$. This implies that y^* satisfies $-y(S_{123}) \leq -1$. Add to this (i) the assignment constraints on clients j_1, j_2 and j_3, (ii) the VUBs for the client j_1 and the facilities in S_{12} and S_{13}, (iii) the VUBs for the client j_2 and the facilities in S_{12} and S_{23}, and (iv) the VUBs for the client j_3 and the facilities in S_{13} and S_{23}. The resulting inequality implies the 3-client inequality. □

Lemma 3 *If a 3-client inequality is violated, then there exists a most-violated inequality such that all of the x variables appearing on the left-hand side of the inequality are positive at (x^*, y^*).*

Proof. Suppose that a 3-client inequality is violated, and $x^*(i, j_1) = 0$ for some $i \in S_{12}$. Due to the VUBs, $x^*(i, j_2) \leq y_i^*$. Thus, we can remove i from S_{12} without decreasing the violation of the inequality. By symmetry, the same argument applies for j_2, j_3, S_{13} and S_{23}.

Now suppose that $x^*(i, j_1) = 0$ for some $i \in S_{123}$. If $x^*(i, j_2) + x^*(i, j_3) \leq y_i^*$, we can delete i from S_{123} without decreasing the violation of the inequality. Otherwise, we can move i from S_{123} to S_{23} without decreasing the violation of the inequality. A similar argument applies to j_2 and j_3. □

Lemma 4 *If a 3-client inequality is violated, then there exists a most-violated inequality such that $y_i^* > 0$ for all $i \in S_{12} \cup S_{13} \cup S_{23} \cup S_{123}$.*

Proof. Suppose that a 3-client inequality is violated, and $y_i^* = 0$ for some $i \in S_{12}$. Due to the VUBs, $x^*(i, j_1)$ and $x^*(i, j_2)$ must both be zero. Then, if we delete i from S_{12}, the violation of the inequality is unchanged. A similar argument applies to S_{13}, S_{23} and S_{123}. □

The above three lemmas lead us to define the following three sets:

$$\begin{aligned} E &= \{(i,j) : i \in I,\, j \in J,\, x_{ij}^* > 0,\, y_i^* < 1\} \\ I^* &= \{i \in I : 0 < y_i^* < 1,\, \exists j \in J : x_{ij}^* > 0\} \\ J^* &= \{j \in J : \exists i \in I^* : x_{ij}^* > 0\}. \end{aligned}$$

One can think of I^* and J^* as forming the node sets of a bipartite graph, and E as being the edge set.

We are now ready for the main result in this section.

Theorem 1 *The separation problem for the 3-client inequalities (6) can be solved exactly in $O\big(|J^*|^2\,|E|\big)$ time.*

Proof. Given Lemmas 2 to 4, we can assume w.l.o.g. that (a) $\{j_1, j_2, j_3\} \subseteq J^*$, (b) $S_{12} \cup S_{13} \cup S_{23} \cup S_{123} \subseteq I^*$, and (c) if x_{ij} appears on the left-hand side of the 3-client inequality, then $(i, j) \in E$.

To exploit these facts, we begin by computing, for each client $j \in J^*$, a list of the locations $i \in I^*$ for which $(i, j) \in E$. We let $d(j)$ denote the *degree* of client $j \in J^*$, which is the size of the associated list. This computation takes only $O\big(|E|\big)$ time in total. Once this is done, we can compute the four quantities more quickly for each given client triple $\{j_1, j_2, j_3\}$, by only considering locations that are in the relevant lists. The total time taken over all triples is:

$$\sum_{\{j_1,j_2,j_3\}\subseteq J^*} O\big(d(j_1) + d(j_2) + d(j_3)\big) = O\bigg(|J^*|^2 \sum_{j\in J^*} d(j)\bigg) = O\big(|J^*|^2\,|E|\big)$$

as stated. □

Note that the algorithm presented in the proof of Theorem 1 can return several violated inequalities in a single call.

3.3 Separation for 3-cycle inequalities

Recall (from Subsection 2.1) that the 3-cycle inequalities are a special case of the 3-client inequalities, and always define facets of $P(m, n)$. For what follows, we will find it helpful to have a separation algorithm for the 3-cycle inequalities in particular.

It is not difficult to modify the separation algorithm for odd cycle inequalities, described in [7], to obtain an $O\big(m^2(m + n)\big)$-time separation algorithm for the 3-cycle inequalities. We do not go into details, because we will show that one can do better.

To aid the reader, we write the 3-cycle inequalities explicitly as:

$$\sum_{i=1}^{3} \big(x(s_i, t_i) + x(s_i, t_{i+1})\big) \;\le\; \sum_{i=1}^{3} y(s_i) + 1. \tag{7}$$

We also refer to the sets I^*, J^* and E, and the client degrees $d(j)$, that were defined in the previous subsection.

We will use the following three lemmas.

Lemma 5 *If a 3-cycle inequality is violated, then (a) $\{s_1, s_2, s_3\} \subseteq I^*$, (b) $\{t_1, t_2, t_3\} \subseteq J^*$, and (c) if x_{ij} appears on the left-hand side of the inequality, then $(i, j) \in E$.*

Proof. Similar to Lemmas 2 to 4. □

Lemma 6 *Suppose we construct a graph, say $\tilde{G}$, with vertex set I^* and edge set $\tilde{E}$, as follows. An edge $\{i, i'\}$ is included in $\tilde{E}$ if and only if there exists at least one client $j \in J^*$ such that both (i, j) and (i', j) lie in E. The weight of the edge $\{i, i'\}$ is set to:*

$$\min_{j \in J^*} \Big\{ 1 + y^*(i) + y^*(i') - 2x^*(i, j) - 2^* x(i', j) \Big\}. \tag{8}$$

Then, a violated 3-cycle inequality exists if and only if there exists in $\tilde{G}$ a triangle with total weight less than 1.

Proof. The 3-cycle inequality can be written in the form:

$$\sum_{k=1}^{3} \Big(1 + y(s_k) + y(s_{k+1}) - 2x(s_k, t_{k+1}) - 2x(s_{k+1}, t_{k+1}) \Big) \ \geq \ 1.$$

The result then follows from Lemma 5. □

Lemma 7 *If a 3-cycle inequality is violated, then at least one of the six x variables appearing on the left-hand side must take a value larger than $1/3$ at (x^*, y^*).*

Proof. Let LHS denote the left-hand side of (7). Suppose that all six x variables take a value less than or equal to $1/3$. Then (x^*, y^*) satisfies the inequality LHS ≤ 2. Now, summing together the VUBs for all six x variables, we obtain the inequality LHS $\leq 2 \sum_{i=1}^{3} y(s_i)$. These two inequalities imply the 3-cycle inequality. □

Now, for each edge $\{i, i'\} \in \tilde{E}$, let $j(i, i')$ denote the client achieving the minimum in (8). We call the edge $\{i, i'\}$ "promising" if at least one of $x^*\big(i, j(i, i')\big)$ and $x^*\big(i', j(i, i')\big)$ exceeds $1/3$. By Lemma 7, any triangle in $\tilde{G}$ that yields a violate 3-cycle inequality must contain at least one promising edge. We also have the following lemma:

Lemma 8 *The number of promising edges in $\tilde{E}$ is $O(|E|)$.*

Proof. For a given client $j \in J^*$, there can exist at most two x variables that exceed $1/3$, due to the assignment constraints. So the number of promising edges involving client j is less than $2d(j)$, and the total number of promising edges is less than $2|E|$. □

We can now present our main result for the 3-cycle inequalities.

Proposition 2 *The separation problem for the 3-cycle inequalities can be solved in $O\big(|I^*|\,|E|\big)$ time.*

Proof. We first show that $\tilde{E}$ can be constructed in $O\big(|I^*|\,|E|\big)$ time. For each client $j \in J^*$, there are $d(j)$ edges in E. Thus, one need check only $\binom{d(j)}{2}$ facility pairs $\{i, i'\}$ for the given client j. Each such pair should be included in $\tilde{E}$. The total time needed to construct $\tilde{E}$ is

$$\sum_{j \in J^*} O\big(d^2(j)\big) = |I^*| \sum_{j \in J^*} O\big(d(j)\big) = O\big(|I^*|\,|E|\big)$$

Note that we can update the weights of the edges in $\tilde{E}$ as we go along.

To complete the proof, we must show that one can check all triangles efficiently. To do this, one simply takes each of the $O(|E|)$ promising edges, and each of the $O(|I^*|)$ candidates for the third facility, and checks if the sum of the three associated edges is less than 1. □

We remark that this algorithm can return several violated inequalities in a single call.

4 Computational Experiments

In this section, we perform computational experiments, to see whether the 3-client inequalities are useful in practice.

4.1 Experimental setup

For a given SPLP instance, we begin by solving the initial LP relaxation, which is obtained from (1)-(4) by replacing the constraints (4) with the trivial bounds. Since this LP relaxation is highly degenerate, we use an interior-point method rather than the simplex method. To be specific, we use the `Barrier` solver of the `CPLEX` callable library (v. 12.10).

After the initial relaxation has been solved, we run a cutting-plane algorithm that uses the separation routines described in Subsections 3.2 and 3.3. To re-optimise the LP after adding cuts, we use a dual simplex algorithm: the `DualOpt` solver of the `CPLEX` callable library. For more details, see Algorithm 1.

All code was written in C++ and compiled using g++ with -O3 optimisations. All experiments were run on a 2.299 GHz AMD Opteron 6376 with 16Gb RAM under a 64 bit Linux operating system (Ubuntu 20.04).

4.2 Test instances

Several sets of benchmark instances are available in the `UFLlib` [29]. For some instances, such as the planar Euclidean ones, the lower bound from the initial LP relaxation is already very close to the optimum. This led us to select the following sets of instances, each of which has an average integrality gap of more than 1%:

Algorithm 1: Cutting-Plane Algorithm for the SPLP

input : Number of facilities n, number of clients m,
facility costs f_i, assignment costs c_{ij}

Solve initial LP (with Barrier);
Record the lower bound LB1;
`// 3-cycle phase`
repeat
  Set `cuts-found` to *false*;
  Call separation algorithm for 3-cycle inequalities;
  if *any 3-cycle inequalities are violated by more than* 10^{-4} **then**
    Add the inequalities to the LP;
    Set `cuts-found` to *true*;
  end
  if `cuts-found` = *true* **then**
    Re-optimise LP (with DualOpt);
    Delete 3-cycle inequalities with slack > 0.1;
  end
until `cuts-found` = *false*;
Record the lower bound LB2;
`// 3-client phase`
repeat
  Set `cuts-found` to *false*;
  Call separation algorithm for 3-client inequalities;
  if *any 3-client inequalities are violated by more than* 10^{-4} **then**
    Add the inequalities to LP;
    Set `cuts-found` to *true*;
  end
  if `cuts-found` = *true* **then**
    Re-optimise LP (with DualOpt);
    Delete 3-client inequalities with slack > 0.1;
  end
until `cuts-found` = *false*;
Record the lower bound LB3;
output: Lower bounds LB1, LB2 and LB3

- The `BK` instances. These were created following the scheme described in Bilde & Krarup [4]. There are 10 instances of types "B" and "C", and 100 instances of types "D" and "E", making 220 in total. The c_{ij} are random integers in the range $[0, 1000]$. The f_i are random for the B and C instances, but constant for the D and E instances. These instances are fairly easy to solve exactly.

- The `M*` instances of Kratica *et al.* [24]. We selected the instances with $m = n \in \{100, 200, 300, 500\}$, for which optimal solution values are known (e.g., [16, 28]). There are 5 instances of each size, making 20 in total. In these instances, there is a negative correlation between the facility costs and assignment costs.

- The `KG` instances. These were created by Ghosh [18] using a similar scheme to that of Koerkel [20]. They come in two types, symmetric and asymmetric. We selected the ones with $m = n \in \{250, 500\}$, for which optimal solutions are known (e.g., [16, 28]). There are 15 instances of each type and size, making 60 in total.

- Instances from Kochetov & Ivanenko [21], which we call `KI`. They only have $m = n = 100$. The ones of type "A", "B" and "C" are designed to have large integrality gaps, and they are surprisingly challenging for exact methods. The ones of type "U" have costs taken from a uniform distribution. There are 30 instances of each type, making 120 in total. Optimal solutions are known for all of these instances [21].

4.3 Results

As mentioned above, our code generates three lower bounds (LB1, LB2 and LB3). For each instance and each bound, we computed the "percentage gap", by which we mean the difference between the lower bound and the optimum, expressed as a percentage of the optimum. We also computed the percentage of the initial gap that was closed by the 3-cycle and 3-client inequalities.

Table 1 displays the following for each set of instances: the name, the number of facilities m, the number of clients n, the mean percentage gap for each of the three bounds, and the percentage gap closed by the two types of inequalities. We see that the 3-cycle inequalities perform rather poorly, closing a significant amount of gap only for the `BK` instances of type B. The 3-client inequalities, on the other hand, close a substantial portion of the gap for all of the `BK` and `M*` instances, and do reasonably well on the uniform `KI` instances.

Table 2 displays, for each set of instances, the mean time (in seconds) taken to compute each the three lower bounds. It is apparent that using 3-cycle inequalities leads to only a modest increase in running time. On the

			% gap			% gap closed	
Set	m	n	LB1	LB2	LB3	3-cycle	3-client
BK-B	30	80	1.87	1.04	0.07	51.27	78.38
BK-C	50	100	6.08	5.75	3.86	7.72	42.85
BK-D	30	80	7.16	7.08	2.82	2.18	64.05
BK-E	50	100	9.19	9.15	5.64	0.65	39.21
MO	100	100	2.85	2.69	0.91	6.21	71.19
MP	200	200	4.20	4.11	2.16	2.90	52.25
MQ	300	300	4.04	4.01	2.20	0.81	46.87
MR	500	500	6.52	6.50	4.83	0.37	26.56
KG-S	250	250	1.46	1.45	1.24	4.89	13.33
KG-S	500	500	1.36	1.36	1.32	0.90	2.74
KG-A	250	250	1.41	1.41	1.09	4.79	13.76
KG-A	500	500	1.36	1.36	1.32	1.48	3.38
KI-A	100	100	25.63	25.63	25.21	0.01	1.63
KI-B	100	100	21.10	20.93	20.54	0.79	2.65
KI-C	100	100	28.15	28.15	28.08	0.00	0.25
KI-U	100	100	4.64	3.91	3.43	17.00	27.90

Table 1: Average percentage gap and percentage gap closed

Set	m	n	LB1	LB2	LB3
BK-B	30	80	0.09	0.14	0.80
BK-C	50	100	0.12	0.18	3.23
BK-D	30	80	0.11	0.12	37.89
BK-E	50	100	0.12	0.13	146.25
MO	100	100	0.45	0.67	55.92
MP	200	200	2.32	5.42	972.55
MQ	300	300	7.62	15.15	7606.92
MR	500	500	34.10	86.95	36077.78
KG-S	250	250	5.24	6.21	2394.32
KG-S	500	500	38.20	43.44	1481.43
KG-A	250	250	4.82	5.72	2061.74
KG-A	500	500	38.05	51.86	1627.31
KI-A	100	100	0.22	0.23	1.84
KI-B	100	100	0.25	0.32	1.53
KI-C	100	100	0.20	0.20	1.45
KI-U	100	100	0.40	0.57	1.91

Table 2: Average time taken to compute lower bounds

other hand, using 3-client inequalities turned out to be more time-consuming than we had hoped. As far as we can tell, the main bottleneck was the re-optimisation of the LPs, rather than the separation algorithm itself. This may be due to degeneracy.

We believe that it would be worthwhile devising effective exact and/or heuristic separation algorithms for other families of valid inequalities for the SPLP. It would also be worthwhile designing and testing a full branch-and-cut algorithm for hard SPLP instances.

Acknowledgement: The authors gratefully acknowledge partial financial support from the University of Pisa, under the project "Analisi di reti complesse: dalla teoria alle applicazioni" (grant PRA_2020_61).

References

[1] K. Aardal (1998) Capacitated facility location: separation algorithms and computational experience. *Math. Program.*, 81, 149–175.

[2] S. Ahn, C. Cooper, G. Cornuéjols & A.M. Frieze (1988) Probabilistic analysis of a relaxation for the p-median problem. *Math. Oper. Res.*,

13, 1–31.

[3] M. Balinski (1965) Integer programming: methods, uses, computation. *Mgmt. Sci.*, 12, 254–313.

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

[5] L. Cánovas, M. Landete & A. Marín (2002) On the facets of the simple plant location packing polytope. *Discr. Appl. Math.*, 124, 27–53.

[6] A. Caprara & M. Fischetti (1996) $\{0, \frac{1}{2}\}$-Chvátal-Gomory cuts. *Math. Program.*, 74, 221–235.

[7] A. Caprara & M. Fischetti (1996) Odd cut-sets, odd cycles, and 0-1/2 Chvátal-Gomory cuts. *Ricerca Operativa*, 26, 51–80.

[8] A. Caprara & J.J. Salazar–González (1996) A branch-and-cut algorithm for a generalization of the uncapacitated facility location problem. *Top*, 4, 135–163.

[9] D.-S. Chen, R.G. Batson & Y. Dang (2010) *Applied Integer Programming: Modeling and Solution*. Wiley.

[10] D.C. Cho, E.J. Johnson, M.W. Padberg & M.R. Rao (1983a) On the uncapacitated location problem I: valid inequalities and facets. *Math. Oper. Res.*, 8, 579–589.

[11] D.C. Cho, E.J. Johnson, M.W. Padberg & M.R. Rao (1983b) On the uncapacitated location problem II: facets and lifting theorems. *Math. Oper. Res.*, 8, 590–612.

[12] M. Conforti, G. Cornuéjols & G. Zambelli (2015) *Integer Programming*. Cham, Switzerland: Springer.

[13] G. Cornuéjols, M.L. Fischer & G.L. Nemhauser (1977) On the uncapacitated location problem. *Ann. Discr. Math.*, 1, 163–177.

[14] G. Cornuéjols, G.L. Nemhauser & L.A. Wolsey (1990) The uncapacitated facility location problem. In: P.B. Mirchandani & R.L. Francis (eds.), *Discrete Location Theory*, pp. 1–54. New York: Wiley.

[15] G. Cornuéjols & J.-M. Thizy (1982) Some facets of the simple plant location polytope. *Math. Program.*, 23, 50–74.

[16] M. Fischetti, I. Ljubić, & M. Sinnl (2017) Redesigning Benders decomposition for large-scale facility location. *Mgmt. Sci.*, 63, 2146–2162.

[17] L. Galli, A.N. Letchford & S.J. Miller (2018) New valid inequalities and facets for the simple plant location problem. *Eur. J. Oper. Res.*, 269, 824–833.

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

[19] M. Guignard (1980) Fractional vertices, cuts and facets of the simple plant location problem. *Math. Program. Study*, 12, 150–162.

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

[21] Y. Kochetov & D. Ivanenko (2005) Computationally difficult instances for the uncapacitated facility location problem. In T. Ibaraki, K. Nonobe & M. Yagiura (eds.) *Metaheuristics: Progress as Real Problem Solvers*, pp. 351–367. Boston, MA: Springer.

[22] B. Korte & J. Vygen (2012) *Combinatorial Optimization: Theory and Algorithms* (5th Edn.) Algorithms and Combinatorics vol. 21. Springer.

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

[24] J. Kratica, D. Tosic, V. Filipovic & I. Ljubic (2001) Solving the simple plant location problem by genetic algorithm. *RAIRO Oper. Res.*, 35, 127–142.

[25] M. Labbé & F. Louveaux (1997) Location problems. In: M. Dell'Amico, F. Maffoli & S. Martello (eds.), *Annotated Bibliographies in Combinatorial Optimization*, pp. 261-281. Chichester: Wiley.

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

[27] J.G. Morris (1978) On the extent to which certain fixed-charge depot location problems can be solved by LP. *J. Oper. Res. Soc*, 29, 71–76.

[28] M. Posta, J. Ferland & P. Michelon (2014) An exact cooperative method for the uncapacitated facility location problem. *Math. Program. Comput.*, 6, 199–231.

[29] UflLib: a library of instances of the uncapacitated facility location problem. Available at
`http://www.mpi-inf.mpg.de/departments/d1/projects/benchmarks/UflLib/`

[30] V. Verter (2011) Uncapacitated and capacitated facility location problems. In: H.A. Eiselt & V. Marianov (eds.), *Foundations of Location Analysis*, pp. 25–38. Berlin: Springer.