Integrating cut–and–solve and semi–Lagrangean based dual ascent for the single source capacitated facility location problem

Sune Lauth Gadegaard[a,*]

[a]*Department of Economics and Business Economics, School of Business and Social Sciences, Aarhus University, Fuglesangs Allé 4, DK-8210 Aarhus V, Denmark.*

April 22, 2016

Abstract

This paper describes how the cut–and–solve framework and semi–Lagrangean based dual ascent algorithms can be integrated in two natural ways in order to solve the single source capacitated facility location problem. The first uses the cut–and–solve framework both as a heuristic and as an exact solver for the semi–Lagrangean subproblems. The other uses a semi–Lagrangean based dual ascent algorithm to solve the sparse problems arising in the cut–and–solve algorithm. Furthermore, we developed a simple way to separate a special type of cutting planes from what we denote the effective capacity polytope with generalized upper bounds. From our computational study, we show that the semi–Lagrangean relaxation approach has its merits when the instances are tightly constrained with regards to the capacity of the system, but that it is very hard to compete with a standalone implementation of the cut–and–solve algorithm. We were, however, able to increase the size of the instances solvable by almost 25 percent compared to methodologies proposed in the literature.

Keywords: capacitated location problem, single source, semi-Lagrangean, cut–and–solve, dual ascent

1. Introduction

The *single source capacitated facility location problem* (SSCFLP) is the problem of opening facilities and allocating each customer's demand to one single open facility, while respecting the capacity of the facilities. An optimal solution is a solution that minimizes the total cost incurred by opening facilities and assigning the demand of the customers to the open facilities. Despite the fact that the problem has a simple verbal description, it is strongly $\mathcal{NP}$–hard and most research has consequently been devoted to heuristics. It is beyond the scope of this paper to review the vast literature on heuristics for the SSCFLP, and we will therefore limit the discussion of heuristics for the SSCFLP to papers dealing with Lagrangean or semi–Lagrangean heuristics.

Klincewicz and Luss (1986) relax the so–called capacity constraints in a Lagrangean manner where the Lagrangean subproblem becomes an uncapacitated facility location problem. Primal solutions are obtained by an add heuristic and a refinement heuristic improving the primal feasible solutions to the Lagrangean subproblem. If the assignment constraints are relaxed in a Lagrangean manner instead, Bitran, Chandru, Sempolinski, and Shapiro (1981) showed that the Lagrangean subproblem decomposes into an independent knapsack problem for each facility site. Sridharan (1993) uses this fact to derive a tight lower bound for the SSCFLP by maximizing the Lagrangean dual function using subgradient optimization. In order to obtain good primal solutions, a generalized assignment problem is solved over the open facilities in each iteration of the subgradient procedure which makes this approach unsuited for large problem instances. The capacitated

[*]Corresponding author.
Email address: `sgadegaard@econ.au.dk` (Sune Lauth Gadegaard)

concentrator location problem, which is equivalent to the SSCFLP, is studied in Pirkul (1987), where a lower bound is obtained by maximizing the Lagrangean dual function resulting when relaxing the demand constraints. In Beasley (1993) both the capacity constraints and the demand constraints are relaxed and different Lagrangean heuristics are compared. Barceló and Casanovas (1984) consider a slight variant of the SSCFLP where the number of open facilities is limited by a given constant K. They propose a Lagrangean heuristic that decomposes into two subproblems; a plant selection and an allocation problem. The heuristic relaxes the demand constraints in a Lagrangean manner and properties of the dual of the LP-relaxation are used to guide a heuristic computing the Lagrangean multipliers. Barceló, Fernández, and Jörnsten (1991) propose a Lagrangean decomposition approach that separates the demand constraints from all other constraints. The Lagrangean subproblem thus decomposes into two subproblems; one being identical to the one obtained when relaxing the demand constraints and one being a simple semi-assignment problem. Due to the integrality property of the latter subproblem, the Lagrangean decomposition bound is no stronger than the Lagrangean bound based on relaxation of the demand constraints. Barceló, Hallefjord, Fernández, and Jörnsten (1990) suggest, to relax the capacity constraints, including a total demand constraint (stating that the total capacity of the open facilities needs to cover the total demand). In addition, they generate simple cover inequalities from the total demand constraint on the fly during subgradient optimization. The added cover inequalities are then relaxed and consequently do not alter the Lagrangean subproblem, but may nevertheless contribute to an improved bound. The assignment constraints are also dualized by Cortinhal and Captivo (2003) and the solutions to the Lagrangean subproblem are transformed into a feasible solution by a repair heuristics based on local search. A more recent Lagrangean relaxation based heuristic was proposed in Chen and Ting (2008), where the Lagrangean dual bound is approximated using subgradient optimization techniques and a multiple ant colony heuristic is used in each iteration in order to improve the best upper bound. We would also like to mention the heuristic dual ascent method based on a semi–Lagrangean relaxation for the *uncapacitated facility location problem* proposed in Monabbati (2014).

In addition to the computational complexity of the SSCFLP, the traditional arc–based formulation of the problem suffers from a weak LP–bound when the cost of opening facilities is large compared to the assignment costs. This is often the case for instances proposed in the literature and exact approaches have therefore traditionally relied on stronger lower bounds. Neebe and Rao (1983) and Díaz and Fernández (2002) therefore reformulate the problem in a Dantzig–Wolfe fashion and use a branch–and–price algorithm for solving the SSCFLP. Holmberg, Rönnqvist, and Yuan (1999) imbed a Lagrangean relaxation in a branch–and–bound framework and use a repeated matching heuristic to generate upper bounds. Recently, Yang, Chu, and Chen (2012) proposed a cut–and–solve method based on relaxing only assignment variables, effectively using the, also $\mathcal{NP}$–hard, capacitated facility location problem (CFLP) as a relaxation of the SSCFLP. In Gadegaard, Klose, and Nielsen (2016b) an improved cut–and–solve algorithm running in three phases was developed.

Recently, several papers have dealt with the so–called semi–Lagrangean relaxation approach for facility location problems. The approach was proposed in Beltran, Tadonki, and Vial (2006) for the p–median problem, and applied again to the uncapacitated facility location problem (UFLP) in Beltran-Royo, Vial, and Alonso-Ayuso (2012). Beltran et al. (2006) show that the semi–Lagrangean relaxation shows no duality gap, and that it therefore constitute an exact solution procedure. Both Monabbati (2014) and Jörnsten and Klose (2015) propose a surrogate relaxation of a set of constraints before dualizing them in a Lagrangean manner. This *condensed semi–Lagrangean relaxation* does not show any duality gap either, and the resulting Lagrangean dual problem becomes a one–dimensional optimization problem. Based on this observation Monabbati (2014) proposes a heuristic solution procedure from the (UFLP) and Jörnsten and Klose (2015) develop an exact dual ascent algorithm guided by primal feasible solutions.

The semi–Lagrangean approaches proposed in the literature have only been applied to location problems where primal feasible solutions can easily be recovered from the solutions to the semi–Lagrangean dual subproblem. These primal feasible solutions have then been used to guide the search for optimal multipliers. For the SSCFLP it is, however, an $\mathcal{NP}$–hard problem just to find a feasible solution, implying that guiding a dual ascent by feasible solutions can be prohibitive. In addition, the solution approaches for the SSCFLP proposed in the literature often use knapsack separation exclusively from the capacity constraints, thereby ignoring the so called assignment constraints when generating cutting planes.

e wish to close the gap in existing literature by overcoming the above-mentioned obstacles and we contribute as follows:

1. We propose to separate a special kind of cutting planes from what we denote the *effective capacity polytope with generalized upper bounds* in order to strengthen the programs before applying the algorithms.
2. We develop a semi–Lagrangean based dual ascent algorithm which integrates the cut–and–solve framework, both as a heuristic and as an exact solver.
3. We propose an enhanced cut–and–solve algorithm that integrates the semi–Lagrangean based dual ascent algorithm as a solution procedure for the otherwise time consuming subproblems.
4. We test three different algorithms and three different schemes for initializing the semi-Lagrangean multiplier. We show empirically that the semi–Lagrangean based dual ascent algorithms performs very well on instances with a small ratio between the total capacity and the total demand.

The remainder of the paper is organized as follows: Section 2 introduces the semi–Lagrangean based dual ascent and the cut–and–solve algorithms for general integer linear programs. In Section 3 we show how the two aforementioned solutions procedures can be integrated while Section 4 adapts the solution methodologies to the single source capacitated facility location problem. Section 5 reports on extensive computational experiments, and finally we conclude on our findings in Section 6.

2. Preliminaries

In this section we introduce the preliminaries for the remainder of the paper. In Section 2.1 we introduce the concept of semi–Lagrangean relaxation and describe two simple dual ascent algorithms based on this relaxation. Section 2.2 introduces the cut–and–solve framework. To that end, consider the generic integer linear program

$$\begin{aligned} Z^* = \min\ & c^T x \\ \text{s.t.: } & Ax = b \\ & x \in P \cap \{0,1\}^n \end{aligned} \tag{1}$$

where $P \subseteq \mathbb{R}^n_+$ is a polyhedron, A is a rational $m \times n$ matrix, and b and c are rational vectors of dimensions m and n, respectively. Unless otherwise stated, it will be assumed throughout the paper that (1) has an optimal solution.

2.1. Semi–Lagrangean relaxation

In traditional Lagrangean relaxation approaches one would introduce *Lagrangean multipliers* λ and relax the constraints $Ax = b$ in a Lagrangean manner in order to obtain a lower bound of (1) given by $L(\lambda) = \lambda^T b + \min\{(c - \lambda^T A)x : x \in P \cap \{0,1\}^n\}$. Although the traditional approach in many cases improves the bound compared to the LP relaxation bound, the duality gap is usually not closed by maximizing $L(\lambda)$. However, another approach called semi–Lagrangean relaxation was recently proposed in Beltran et al. (2006). By stating the equations $Ax = b$ as the two sets of inequalities $Ax \leq b$ and $Ax \geq b$ we obtain an equivalent formulation of (1). Relaxing the latter set of inequalities with multipliers $\lambda \in \mathbb{R}^m_+$ in a Lagrangean manner results in the *semi–Lagrangean relaxation*

$$\begin{aligned} L(\lambda) = \lambda^T b + \min\ & (c - \lambda^T A)^T x \\ \text{s.t.: } & x \in \mathcal{X}, \end{aligned} \tag{2}$$

where $\mathcal{X} = \{x \in P \cap \{0,1\}^n : Ax \leq b\}$. The Lagrangean dual problem then becomes the problem of finding multipliers $\lambda^* \in \mathbb{R}^m_+$ such that

$$\lambda^* \in \arg\max\{L(\lambda) : \lambda \in \mathbb{R}^m_+\}. \tag{3}$$

As the variables are positive, the system $Ax = b$ can also be described by the inequalities $Ax \leq b$ and the surrogate relaxation $e^T Ax \geq e^T b$ of the constraints $Ax \geq b$, where the vector $e \in \mathbb{R}^m$ is given by $e = (1, \ldots, 1)$. Dualizing the single constraint $e^T Ax \geq e^T b$ with multiplier $\mu \geq 0$ leads to the so–called *condensed semi–Lagrangean relaxation* given by

$$\begin{aligned} \mathcal{L}(\mu) = \mu e^T b + \min\ & (c - \mu e^T A)^T x \\ \text{s.t.:}\ & x \in \mathcal{X}. \end{aligned} \tag{4}$$

The condensed semi–Lagrangean relaxation was suggested by Monabbati (2014) and Jörnsten and Klose (2015). Compared to the semi–Lagrangean dual problem (3) the dual problem

$$\max\{\mathcal{L}(\mu)\ :\ \mu \geq 0\}, \tag{5}$$

has the clear advantage that it is one–dimensional. Preliminary tests confirmed the findings for the uncapacitated facility location problem reported in Jörnsten and Klose (2015), namely that the condensed semi–Lagrangean relaxation outperforms the ordinary semi–Lagrangean relaxation by an order of magnitude. Therefore this paper is limited to the study of the condensed semi–Lagrangean relaxation only. For that reason, we will henceforth use the term *semi–Lagrangean relaxation* for the relaxation (4) instead of the rather cumbersome *condensed semi–Lagrangean relaxation.*

In Theorem 1 we have stated some properties of the semi–Lagrangean function and the semi–Lagrangean dual problem which were originally proven in Monabbati (2014).

Theorem 1 (Monabbati (2014))**.** *Let $\mathcal{X}(\mu) = \arg\min\{(c - \mu e^T A)x\ :\ x \in \mathcal{X}\}$ and let $\mathcal{M} = \arg\max\{\mathcal{L}(\mu)\ :\ \mu \geq 0\}$. Then the following holds true*

1. *$\mathcal{L}(\mu)$ is monotone and $\mathcal{L}(\mu') \geq \mathcal{L}(\mu)$ if $\mu' \geq \mu$. The inequality is strict if $\mu' > \mu$ and $\mu' \notin \mathcal{M}$.*
2. *If $x^* \in \mathcal{X}(\mu)$ and $Ax^* = b$, then $\mu \in \mathcal{M}$ and x^* is optimal for* (1)*.*
3. *Conversely, if $\mu^* \in int(\mathcal{M})$, then every $x \in \mathcal{X}(\mu^*)$ is optimal for* (1) *(where $int(\mathcal{M})$ denotes the* interior *of the set $\mathcal{M}$).*
4. $\max\{\mathcal{L}(\mu)\ :\ \mu \geq 0\} = Z^*$*, that is, the semi–Lagrangean relaxation shows no duality gap.*

From Theorem 1 point 1 we see that the lower bound produced by $\mathcal{L}(\mu)$ is monotone and increasing in μ, and in point 4 it is stated that the dual problem (5) closes the duality gap. This obviously implies that if μ^* is an optimal dual multiplier, then every $\mu \geq \mu^*$ will also be optimal. Furthermore, point 2 states that if an optimal solution to the dual problem is primal feasible, then it is primal optimal and the corresponding dual multiplier is optimal for the semi–Lagrangean dual problem. On the other hand, if $\mu \in int(\mathcal{M})$, then an optimal solution to the semi–Lagrangean subproblem will be a primal optimal solution as well. Theorem 1 naturally leads to the generic dual ascent algorithm described in Algorithm 1. In Step 1, the multiplier μ is initialized. Next, the semi–Lagrangean subproblem is solved to optimality in Step 2. The solution obtained in Step 2 is tested for primal feasibility in Step 3. According to point 2 of Theorem 1 the solution x^k to the semi–Lagrangean subproblem is optimal to (1) if x^k is primal feasible, and the procedure can therefore be terminated. But if $Ax^k \neq b$, the multiplier μ is incremented in Step 4 and the algorithm returns to Step 2.

The obvious drawback of this methodology is that the semi–Lagrangean subproblem has the same computational complexity as the original problem, meaning that the subproblem might be just as hard to solve as the original problem. One should, however, note that if the vector x can be decomposed in such a way that if $x = (x^1, x^2) \in \mathcal{X}$ and $\tilde{x}^1 \leq x^1$ then $(\tilde{x}^1, x^2) \in \mathcal{X}$, then the semi–Lagrangean subproblem (2) can be reduced in size by eliminating all variables x_j^1 where $c_j^1 - \mu e^T a_j^1 \geq 0$. Here a_j^1 is the column corresponding to the variable x_j^1. Also note that the larger the value of μ, the fewer variables can be eliminated in this way. On the other hand, it should be obvious that for a sufficiently large value of μ the solution to the semi–Lagrangean subproblem will be optimal to (1). Thus, one should find a $\mu \in int(\mathcal{M})$ which is as small as possible and if possible, find such a multiplier using a method solving the semi–Lagrangean subproblem as a few times as possible.

```
Input: Vectors $c$ and $b$, matrix $A$, and feasible set $\mathcal{X}$.
Output: An optimal solution $x^*$ to (1) and optimal multiplier $\mu^* \in \mathcal{M}$.

Step 1: (Initialization) Set $k = 1$ and initialize multiplier, $\mu^k$, to a suitably small value.

Step 2: (Solve subproblem) Obtain an optimal solution $x^k$ to the semi–Lagrangean subproblem

        $\mathcal{L}(\mu) = \min\{(c - \mu^k e^T A)^T x \; : \; x \in \mathcal{X}\}.$

Step 3: (Termination check) If $Ax^k = b$, stop with $(x^k, \mu^k)$ as an optimal primal–dual pair.

Step 4: (Update) Let $\mu^{k+1} = \mu^k + \delta^k$, $\delta^k > 0$, and return to Step 2. Go to Step 2.
```

Algorithm 1: A dual ascent algorithm for a semi–Lagrangean relaxation of a general ILP.

```
Input: Vectors $c$ and $b$, matrix $A$, and feasible set $\mathcal{X}$.
Output: An optimal solution $x^*$ to (1) and optimal multiplier $\mu^* \in \mathcal{M}$.

Step 1: (Initialization) Set $k = 1$ and initialize multiplier, $\mu^k$, to a suitably small value.

Step 2: (Solve subproblem)

    Step 2.1: Use a heuristic to obtain a good feasible solution $\bar{x}^k$ to

            $\min\{(c - \mu^k e^T A)^T x \; : \; x \in \mathcal{X}\}.$        (6)

        If $A\bar{x} \neq b$, set $x^k = \bar{x}^k$ and go to Step 3. Otherwise, the subproblem must be solved to
        optimality, therefore go to Step 2.2.

    Step 2.2: Determine $x^k \in \mathcal{X}(\mu^k)$, that is an optimal solution to (6), and proceed to Step 3.

Step 3: (Termination check) If $Ax^k = b$, stop with $(x^k, \mu^k)$ as an optimal primal–dual pair.

Step 4: (Update) Let $\mu^{k+1} = \mu^k + \delta^k$, $\delta^k > 0$, and return to Step 2. Go to Step 2.
```

Algorithm 2: A dual ascent algorithm with heuristically solved subproblems for a semi–Lagrangean relaxation of a general ILP.

Just like pricing problems can be solved heuristically in a column generation procedure, the semi–Lagrangean subproblem can also be solved heuristically to indicate optimality of the multiplier μ. This leads to Algorithm 2, which is a version of Algorithm 1 where Step 2 is expanded into two sub–steps that allow the subproblem to be solved heuristically. In Algorithm 2 the semi–Lagrangean subproblem is only solved to optimality if the heuristic suggests that the multiplier μ^k is optimal ($A\bar{x}^k = b$). Otherwise the subproblem is simply solved by a heuristic that is assumed to produced good (near optimal) solutions in reasonable time. This can potentially speed ud the search for optimal dual multipliers by reducing the time spent on solving the subproblems. On the other hand, this might increase the value of μ more than necessary, implying that the final subproblem cannot be reduced as much as otherwise possible.

2.1.1. Updating the multiplier

In Step 4 of Algorithms 1 and 2 the semi–Lagrangean multiplier is updated by adding a strictly positive number to the current multiplier. One possible way of updating the multiplier is to use subgradient optimization. Let x^k be an *optimal* solution to the semi–Lagrangean subproblem at multiplier μ^k. Then

$$g^k = e^T(b - Ax^k)$$

is easily seen to be a subgradient at μ^k. The multiplier of iteration $k+1$ can then be determined as $\mu^{k+1} = \mu^k + \phi^k g^k$, where $0 < \phi^k$ is a positive step size. A step size often suggested in the literature on subgradient optimization is

$$\phi^k = \frac{UB - \mathcal{L}(\mu^k)}{(e^T(b - Ax^k))},$$

where UB is an upper bound on the optimal solution value of the original problem. This leads to the updating strategy $\mu^{k+1} = \mu^k + UB - \mathcal{L}(\mu^k)$. One might, however, also regularize the step size by the length of the gradient whereby the new iterate is determined by $\mu^{k+1} = \mu^k + \frac{UB - \mathcal{L}(\mu^k)}{e^T(b-Ax^k)}$. Both of these strategies close the duality gap as the increment is strictly positive until the duality gap is closed. The main drawback of this method is that the semi–Lagrangean subproblem needs to be solved to optimality for g^k to be a subgradient. Another and more straightforward procedure is to simply use a strictly positive $\delta^k = \delta > 0$ and increment the multiplier μ^k by a fixed amount in each iteration. In Section 4.2 we propose an updating scheme which is based on dual feasible solutions.

2.2. Cut–and–solve

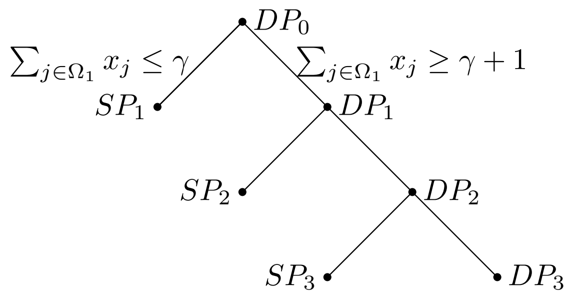
The cut–and–solve approach was first proposed by Climer and Zhang (2006) for the asymmetric traveling salesman problem. The methodology is essentially a branch–and–bound algorithm which instead of branching on a single variable branches on a *set* of variables. At each level in the search tree there are only two nodes (see Figure 1). The left node is associated with a linear constraint stating that the sum of the variables in a set Ω is less than or equal to an integer γ, while the right node is associated with the sum being larger than or equal to $\gamma + 1$. The MIP corresponding to a left node is called a *sparse problem* (SP) as the search space is considerably reduced compared to the original problem. The problems solved at the right nodes are called dense problems and the cut $\sum_{i \in \Omega} x_i \geq \gamma + 1$ is called a *piercing cut.* For MIPs where the integer constrained variables are binary, it is natural to select the constant $\gamma = 0$. In this way, all variables in the set Ω will be fixed to zero in the sparse problem, thus reducing the problem significantly.

A generic cut–and–solve algorithm for the MIP (1) is given in Algorithm 3. In Step 1, the lower and upper bounds are initialized and the set of piercing cuts is initialized to the empty set. In the second step, Step 2, of Algorithm 3 a lower bound of the dense problem is obtained. Note that this lower bound is a lower bound on the optimal solution in the space *not* removed by the piercing cuts. In Step 3, the lower bound is compared to the current best solution and if the lower bound exceeds the value of the incumbent, the incumbent is optimal. In Step 4, a piercing cut is selected and in Step 5, the SP associated with the piercing cut is solved. If the solution to SP improves the current incumbent, it is updated in Step 6 and the value of the incumbent is compared to the lower bound obtained in Step 1. If the lower bound exceeds the new incumbent, an optimal solution has been found. Finally, in Step 7, the search space investigated in the sparse problem in Step 5 is excluded by the addition of the piercing cut and the procedure is repeated from Step 2.

Climer and Zhang (2006) propose to use the LP–relaxation of the dense problem and to define the set Ω as those variables having a reduced cost above a specified (positive) threshold. Climer and Zhang (2006) prove termination of the cut–and–solve algorithm, Algorithm 3, by assuming that the search space removed by the piercing cuts removed at least one feasible solution from the problem. It was noted in Gadegaard et al. (2016b) that an optimal solution to the original problem was often found at the first sparse problem or in nodes close to the top of the search tree when the sparse problems are chosen carefully. Thus, if the cut–and–solve algorithm is stopped before optimality is proven, it becomes a heuristic with a known quality as both a lower and an upper bound of the problem are provided by the algorithm. If the sparse problems are chosen appropriately, an optimal solution might be found very close to the top of the search tree, yielding a potentially powerful heuristic.

3. Integrating cut–and–solve and semi–Lagrangean based dual ascent

In this section we describe two ways of integrating the two frameworks of cut–and–solve and semi–Lagrangean based dual ascent. There are two natural ways to integrate these two frameworks; the first



DP_0
$\sum_{j\in\Omega_1} x_j \leq \gamma$
$\sum_{j\in\Omega_1} x_j \geq \gamma + 1$
SP_1
DP_1
SP_2
DP_2
SP_3
DP_3

Figure 1: The cut–and–solve search tree.

Input: A mixed inter programming problem.
Output: An optimal solution x^* to (1).

<u>*Step 1:*</u> (Initialization) Set $L = -\infty$, $UB = \infty$, and $H = \emptyset$, where H is the set of piercing cuts.

<u>*Step 2:*</u> (Dense problem) Solve a relaxation of (1) with all piercing cuts in H added. Let the solution value be L.

<u>*Step 3:*</u> (Termination check) If $L \geq UB$, return the incumbent.

<u>*Step 4:*</u> (Piercing cut selection) Select an index set, Ω, of variables for the piercing cut.

<u>*Step 5:*</u> (Sparse problem) Solve the problem (1) with the constraint $\sum_{j\in\Omega} x_j \leq \gamma$ added.

<u>*Step 6:*</u> (Incumbent update) If the solution found in Step 4 improves the incumbent, then update UB. If $L \geq UB$, return the incumbent.

<u>*Step 7:*</u> (Adding piercing cut) Add piercing cut $\sum_{j\in\Omega} x_j \geq \gamma + 1$ to H and go to Step 1.

Algorithm 3: A description of a generic cut–and–solve algorithm.

approach is to solve the subproblems arising in the semi–Lagrangean dual ascent algorithm using the cut–and–solve framework. Section 3.1 proposes a way to do this in which the cut–and–solve framework is used as a heuristic and only as an exact solver when the heuristic solution to the semi–Lagrangean subproblems is primal feasible. The second natural way is to solve the sparse problems arising in the cut–and–solve algorithm using a semi–Lagrangean dual ascent algorithm. Such an approach is developed in Section 3.2.

3.1. Integrating cut–and–solve in a semi–Lagrangean based dual ascent algorithm

As mentioned in Section 2.2, the cut–and–solve framework can be used both as a heuristic and as an exact solution method. This makes the cut–and–solve framework ideal as a method to perform both Steps 2.1 and 2.2 in Algorithm 2. If the cut–and–solve algorithm described in Algorithm 3 is terminated before optimality has been proven, the framework becomes a heuristic. This means that Step 2.1 of Algorithm 2 can be handled by a truncated version of the cut–and–solve algorithm, while Step 2.2, where an optimal solution is found to the semi–Lagrangean subproblem, can be handled by *continuing* from the cut–and–solve branching node where the heuristic stopped. Such a framework is proposed in Algorithm 4.

In this setting, the cut–and–solve framework (Steps 2.1 to 2.7 of Algorithm 4) is used as a heuristic. The search is terminated as soon as a primal infeasible solution that improves the incumbent of the subproblem is found. Only if no such solution to the sparse problems is found will the semi–Lagrangean subproblem be solved to optimality. Note that the choice of piercing cuts may make the sparse problems infeasible. The proof of termination of the cut–and–solve algorithm provided by Climer and Zhang (2006) rely on the sparse problems containing at least one feasible solution, meaning that the piercing cuts have to be chosen accordingly or that another termination proof must be provided. If the piercing cuts are chosen such that the sparse problems always exhibit a feasible solution, the modified cut–and–solve algorithm in Steps 2.1 to 2.7 of Algorithm 4 only performs one iteration unless the solution found is primal feasible. Therefore, in the first iterations of the dual ascent algorithm, little computational effort is put into solving the subproblems. Only when the top nodes of the cut–and–solve search tree start to indicate that the multiplier μ^k might be optimal (by finding primal feasible solutions), the effort is increased.

```
Input: Vectors $c$ and $b$, matrix $A$, and feasible set $\mathcal{X}$.
Output: An optimal solution $x^*$ to (1) and optimal multiplier $\mu^* \in \mathcal{M}$.

Step 1: (Initialization) Set $k = 1$ and initialize the multiplier, $\mu^k$, to a suitably small value.

Step 2: (Solving subproblem) Use the cut–and–solve framework to solve the semi–Lagrangean subproblem

    Step 2.1: (Initialization of cut–and–solve) Set $best = \infty$ and initialize the incumbent $x^k = null$.
    Step 2.2: (Solution of dense problem) Solve a relaxation of
        $$\min\{(c - \mu^k e^T A)^T x \ : \ x \in \mathcal{X}\},$$
        and let $Z^{DP}$ be the objective function value of the relaxation. If $Z^{DP} \geq best$, go to Step 3
        with $x^k$ as an optimal solution to the semi–Lagrangean subproblem. Otherwise, proceed to
        Step 2.3.
    Step 2.3: (Piercing cut selection) Determine a subset of variables, $\Omega$, defining the piercing cut.
    Step 2.4: (Solution of sparse problem) If the sparse problem is feasible, set
        $$Z^{SP} = \min\{(c - \mu^k e^T A)^T x \ : \ x \in \mathcal{X}, \ \sum_{j \in \Omega} x_j \leq \gamma\}.$$
        Else set $Z^{SP} = \infty$. Go to Step 2.5.
    Step 2.5 (Incumbent update) If $Z^{SP} \leq best$, update the subproblem incumbent $x^k$, set $best = Z^{SP}$
        and go to Step 2.6. Else, go to Step 2.7.
    Step 2.6 (Feasibility check) If $Ax^k \neq b$, go to Step 4. Else, go to Step 2.7.
    Step 2.7 (Adding piercing cut) Add the piercing cut $\sum_{j \in \Omega} x_j \geq \gamma + 1$ to $\mathcal{X}$ and go to Step 2.2.

Step 3: (Termination check) If $Ax^k = b$, stop with $(x^k, \mu^k)$ as an optimal primal–dual pair.

Step 4: (Multiplier update) Let $\mu^{k+1} = \mu^k + \delta^k$, $\delta^k > 0$, and return to Step 1. Go to Step 1.
```

Algorithm 4: A semi–Lagrangean based dual ascent algorithm integrating the cut–and–solve framework for solving the semi–Lagrangean subproblems.

3.2. Integrating semi–Lagrangean dual ascent in a cut–and–solve algorithm

When employing the cut–and–solve framework for solving integer programming problems one of the obstacles is to be able to solve the sparse problems in Step 5 of Algorithm 3 efficiently. Even though the piercing cuts reduce the size of the sparse problem considerably compared to the original problem, it might still be very time consuming to solve these reduced problems. The dual ascent algorithm described in Algorithm 2 might offer a way to further reduce the sparse problem and thereby speed up the solution of difficult problems. Furthermore, when the feasibility version of the problem (1) is $\mathcal{NP}$–hard, it can be very hard to find a feasible, and specifically an optimal, solution to the sparse problems. Therefore, relaxing the equation system $Ax = b$ to the inequality system, $Ax \leq b$, can make it easier to find feasible solutions that can be utilized by heuristics in the solver employed to solve the subproblems. For completeness, Algorithm 5 describes the cut–and–solve algorithm using a dual ascent algorithm to solve the sparse problems. Note that Algorithm 5 is identical to Algorithm 3 except for Step 4, where Algorithm 5 uses the semi–Lagrangean based dual ascent algorithm to solve the sparse problems.

4. Applications to the single source capacitated facility location problem

In this section, we start by formulating the single source capacitated facility location problem (SSCFLP). Next, we apply the semi–Lagrangean relaxation to the single source capacitated facility location problem (SSCFLP) and suggest procedures for initializing and updating the semi–Lagrangean dual multiplier, μ. We continue by proposing a new way to generate cutting planes that can be used to strengthen the lower bound of the problems. Finally, we propose a dual ascent algorithm integrating the cut–and–solve framework and a cut–and–solve algorithm integrating the dual ascent algorithm.

The SSCFLP can be formulated as a pure integer linear programming (ILP) problem in the following way: Let $\mathcal{I}$, $|\mathcal{I}| = n$, be a set of potential facility sites and $\mathcal{J}$, $|\mathcal{J}| = m$, be a set of customers. Each facility, $i \in \mathcal{I}$, has a fixed capacity, $s_i > 0$, and each customer has a fixed and known demand, $d_j > 0$. Opening a facility at site $i \in \mathcal{I}$ results in a fixed opening cost $f_i > 0$ and allocation of customer j to facility i yields a cost of $c_{ij} \geq 0$. Let y_i be a binary variable equaling one if and only if a facility is opened at site i and similarly let x_{ij} be a binary variable that equals one if and only if customer j is allocated to facility i. The SSCFLP can then be stated as the following ILP problem:

$$\min \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij}x_{ij} + \sum_{i\in\mathcal{I}} f_i y_i \tag{7}$$

$$\text{s.t.:} \sum_{i\in\mathcal{I}} x_{ij} = 1, \quad \forall j \in \mathcal{J}, \tag{8}$$

$$\sum_{j\in\mathcal{J}} d_j x_{ij} \leq s_i y_i, \quad \forall i \in \mathcal{I}, \tag{9}$$

$$0 \leq y_i,\ x_{ij} \leq 1, \quad \forall i \in \mathcal{I},\ j \in \mathcal{J}, \tag{10}$$

$$y_i,\ x_{ij} \in \{0,1\} \quad \forall i \in \mathcal{I},\ j \in \mathcal{J}. \tag{11}$$

The objective function (7) minimizes the total cost (assignment plus opening costs). Constraints (8) state that each customer needs to be assigned to exactly one facility while constraints (9) make sure that each facility's capacity is respected. Finally, the constraints (10) impose lower and upper bounds on each variable and the constraints (11) state that all variables should be binary.

4.1. Semi–Lagrangean relaxation applied to the SSCFLP

The equations (8) can be replaced by the two sets of inequalities $\sum_{i\in\mathcal{I}} x_{ij} \leq 1$ for all $j \in J$ and $\sum_{j\in\mathcal{J}}\sum_{i\in\mathcal{I}} x_{ij} \geq m$. The latter is then dualized with multiplier $\mu \geq 0$, resulting in the Lagrangean dual

```
Input: Vectors $c$ and $b$, matrix $A$, and feasible set $\mathcal{X}$.
Output: An optimal solution $x^*$ to (1).

Step 1: (Initialization) Set $L = -\infty$, $UB = \infty$, and set the set of piecing cuts $H = \emptyset$. Initialize the
    incumbent to an empty solution $x^* = null$.

Step 2: (Dense problem) Solve a relaxation of (1) with all piercing cuts in $H$ added. Let the solution
    value be $L$.

Step 3: (Termination check) If $L \geq UB$, return the incumbent.

Step 4: (Piercing cut selection) Select a set of variables $\Omega$ for the piercing cut.

Step 5: Use the dual ascent algorithm described in Algorithm 2 to find an optimal solution in the space
    removed by the piercing cut (sparse problem).

    Step 5.1: (Initialization) Set $k = 1$ and initialize multiplier, $\mu^k$, to a suitably small value.
    Step 5.2: (Solve subproblem)
        Step 5.2.1: Use a heuristic to obtain a good feasible solution $\bar{x}^k$ to
            $$\min\{(c - \mu^k e^T A)^T x \ : \ x \in \mathcal{X}, \sum_{j \in \Omega} x_j \leq \gamma\}.$$
            If $A\bar{x} \neq b$, set $x^k = \bar{x}^k$ and go to Step 5.4. Otherwise the subproblem needs to be solved
            to optimality, therefore go to Step 5.2.2.
        Step 5.2.2: Determine $x^k \in \arg\min\{(c - \mu^k e^T A)^T x \ : \ x \in \mathcal{X}, \sum_{j \in \Omega} x_j \leq \gamma\}$, and proceed to
            Step 3.
    Step 5.3: (Termination check) If $Ax^k = b$, go to Step 6 as $x^k$ is an optimal solution to the sparse
        problem. Else go to Step 5.4.
    Step 5.4: (Update) Let $\mu^{k+1} = \mu^k + \delta^k$, $\delta^k > 0$, and return to Step 2. Go to Step 2.

Step 6: (Incumbent update) If the solution found in Step 4 improves the incumbent, then update $UB$.
    If $L \geq UB$, return the incumbent.

Step 7: (Adding piercing cut) Add piercing cut $\sum_{j \in \Omega} x_j \geq \gamma + 1$ to $H$ and go to Step 1.
```

Algorithm 5: A cut–and–solve algorithm integrating a semi–Lagrangean dual ascent algorithm for solving the sparse problems.

function

$$\begin{aligned}\mathcal{L}(\mu) = \mu m + \min\ & \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}}(c_{ij}-\mu)x_{ij} + \sum_{i\in\mathcal{I}} f_i y_i \\ \text{s.t.:}\ & \sum_{i\in\mathcal{I}} x_{ij} \le 1, \quad \forall j\in J, \\ & \sum_{j\in\mathcal{J}} d_j x_{ij} \le s_i y_i, \quad \forall i \in I, \\ & 0 \le x_{ij},\ y_i \le 1, \quad \forall i\in I, j\in J, \\ & x_{ij}, y_i \in \{0,1\}, \quad \forall i\in I, j \in J.\end{aligned}$$

If $(\bar{x},\bar{y})$ is feasible for the Lagrangean subproblem, then $(\tilde{x},\bar{y})$ is feasible too if $\tilde{x}_{ij} \le \bar{x}_{ij}$, $\forall i\in\mathcal{I}, j\in\mathcal{J}$. This implies that we can exclude all variables showing a Lagrangean reduced cost $c_{ij}-\mu \ge 0$ from the subproblem. For small values of μ this may lead to a substantial reduction in the number of x_{ij}–variables.

We can also present the semi–Lagrangean subproblem in the same form as the original problem as follows: Introduce a "dummy" facility, $i=0$, with $c_{0j}=\mu$ for all $j\in\mathcal{J}$, $f_0=0$ and $s_i=D$. Setting $\mathcal{I}_0=\mathcal{I}\cup\{0\}$ the Lagrangean subproblem can be posed as the following SSCFLP

$$\begin{aligned}\mathcal{L}(\mu) = \min\ & \sum_{i\in I_0}\sum_{j\in\mathcal{J}} c_{ij}x_{ij} + \sum_{i\in I_0} f_i y_i \\ \text{s.t.:}\ & \sum_{i\in\mathcal{I}_0} x_{ij} = 1, \quad \forall j\in\mathcal{J}, \\ & \sum_{j\in\mathcal{J}} d_j x_{ij} \le s_i y_i, \quad \forall i\in\mathcal{I}_0, \\ & x_{ij}=0, \quad \forall i\in\mathcal{I}, j\in\mathcal{J} : c_{ij}\ge\mu, \\ & y_0 = 1, \\ & x_{ij},\ y_i\in\{0,1\}, \quad \forall i\in\mathcal{I}_0, j\in\mathcal{J}.\end{aligned} \tag{12}$$

As customer j can be assigned to the "dummy" facility at a cost of μ, all assignment variables x_{ij} with $c_{ij}\ge\mu$ can be removed from the program. This leads to the interpretation of the semi–Lagrangean multiplier as a fixed prize obtainable by servicing a customer. Alternatively, μ can be thought of as a threshold which defines a "core" problem, that gradually has to be enlarged to facilitate the eventual proof of optimality. Core algorithms have proved very effective for problems like the binary knapsack problem (see Pisinger (1997)), but core–like algorithms have also been successfully used as heuristics for facility location problems (see e.g. Avella, Boccia, Sforza, and Vasilév (2008) and Guastaroba and Speranza (2012)).

The lower bound $\mathcal{L}(\mu)$ can be considerably improved by adding the constraint $\sum_{i\in\mathcal{I}} s_i y_i \ge \sum_{j\in\mathcal{J}} d_j := D$ to the semi–Lagrangean subproblem (12). Note that this constraint is redundant for the original problems as it is implied by constraints (8) and (9). However, when the constraints (8) are relaxed in a semi–Lagrangean manner, this is no longer the case. The added constraint ensures that the lower bound $\mathcal{L}(\mu)$ is never worse than $\min\{\sum_{i\in\mathcal{I}} f_i y_i : \sum_{i\in\mathcal{I}} s_i y_i \ge D\} + \sum_{j\in\mathcal{J}} \min_{i\in\mathcal{I}_0} c_{ij}$. Since the semi–Lagrangean subproblem can be cast as an SSCFLP, it can be solved by any algorithm customized for the SSCFLP.

Even though the optimization problem (12) is just as hard as the original SSCFLP (7)–(11) seen from a computational complexity viewpoint, the feasibility version is significantly easier. It is well known that the feasibility problem "does the SSCFLP (7)–(11) contain a feasible solution" is $\mathcal{NP}$-complete (this can for example be seen by reduction from the generalized assignment problem). However, the problem "does the semi–Lagrangean of SSCFLP given by (12) contain a feasible solution" is polynomially solvable, and the answer is always "yes". This means that a feasible solution for (12) is easy to obtain, and that such a feasible solution can be used as a starting solution for a heuristic in order to find a feasible solution to the original problem.

4.2. Initializing and updating the semi–Lagrangean multiplier

It is crucial to have a good initial value for the dual variables for Algorithm 1 to be efficient. For that reason, we start by setting a lower bound on the optimal multiplier.

Proposition 1. *For all $\mu \in \mathcal{M}$ we have $\mu \geq \mu^{\min} := \max_{j \in \mathcal{J}} \min_{i \in \mathcal{I}} c_{ij}$.*

Proof. If there exists a $j \in \mathcal{J}$ such that $\mu < \min_{i \in \mathcal{I}} c_{ij}$, then that customer will be serviced from the dummy facility in any optimal solution to the semi–Lagrangean subproblem. □

For the *uncapacitated* facility location problem Jörnsten and Klose (2015) show that there exists an optimal multiplier with $\mu \leq \max_{i \in \mathcal{I} j \in \mathcal{J}} c_{ij}$. This is, however, not the case for the SSCFLP due to the presence of the capacity constraints. However, a trivial upper bound is readily available by $\mu^{\max} := \sum_{i \in \mathcal{I}} f_i + \max_{i \in \mathcal{I}, j \in \mathcal{J}}\{c_{ij}\}$. This upper bound is obvious, as it is cheaper to open all facilities an assign each customer to its most expensive facility than to assign it to the dummy facility. However, one should note, that if $\mu^k \geq \max\{c_{ij} : i \in \mathcal{I}, j \in \mathcal{J}\}$, then no variables can be eliminated from the subproblem. Below we list four different choices of starting values for the semi–Lagrangean multipliers which all satisfy Proposition 1:

1. Compute (near) optimal Lagrangean multipliers λ^* by for example subgradient methods to the relaxation obtained by relaxing constraints (8) in a Lagrangean manner. Then set $\mu = \min\{\mu^{\max}, \max_{j \in \mathcal{J}} \lceil \lambda_j^* \rceil\}$.
2. Solve the linear relaxation of (7)–(11) and let δ^* be an optimal dual solution corresponding to the assignment constraints (8). Set $\mu = \max_{j \in \mathcal{J}} \lceil \delta_j^* \rceil$.
3. Specify a minimum number of possible assignments for each customer, say T. Then set $\mu = \min\{n \in \mathbb{N} : |\{i \in \mathcal{I} : c_{ij} \leq n\}| \geq T, \forall j \in \mathcal{J}\}$. T should be large enough to yield an "interesting" subproblem.
4. Let $(\bar{x}, \bar{y})$ be a (good) feasible solution for the SSCFLP. Then set $\mu = \min\{c_{ij} : c_{ij} > c(\bar{x}), i \in \mathcal{I}, j \in \mathcal{J}\}$, where $c(\bar{x}) = \max\{c_{ij} : \bar{x}_{ij} > 0\}$.

An obvious disadvantage of the first suggestion for initializing the multipliers is that it requires potentially many subgradient iterations and that a series of knapsack problems needs to be solved in each iteration. The other three suggestions each has their own merits: suggestion 2 is easy to obtain. If in addition cutting planes are used to approximate the convex hull of integer solutions to the knapsack constraints defined by the capacity constraints (9), the optimal dual variables approximate optimal Lagrangean variables as well. Specifying a certain number of allowed assignments in suggestion 3 makes it possible to have complete control over the size of the first subproblem. And finally 4 provides the possibility that improved primal feasible solutions can be found quickly.

The next crucial component regarding the multiplier is the updating scheme. To that end, let (x^k, y^k) be the solution to the semi–Lagrangean subproblem computed in Step 2 of Algorithm 2 at multiplier μ^k. Furthermore, let $\mathcal{J}^0 = \{j \in \mathcal{J} : \sum_{i \in \mathcal{I}} x_{ij}^k = 0\}$. That is, $\mathcal{J}^0$ is the subset of customers which is assigned to the dummy facility. The fact that these customers are effectively not serviced indicates that the prize μ^k is not sufficiently high for these customers to be profitable. Hence, a natural updating rule is

$$\mu^{t+1} = \min\{c_{ij} : c_{ij} > \mu^k, i \in \mathcal{I},\ j \in \mathcal{J}^0\}.$$

However, it is not guaranteed that any $c_{ij} > \mu^k$ for $i \in \mathcal{I}$ and $j \in \mathcal{J}^0$ exists. To overcome this issue, one might try to make some of the already serviced customers even more profitable by updating the multiplier as follows

$$\mu^{k+1} = \min\{c_{ij} : c_{ij} > \mu^k, i \in \mathcal{I},\ j \in \mathcal{J}\}.$$

Even this updating strategy may fail if no assignment cost exceed the value of the current multiplier. This means that no x_{ij}–variables can be removed based on the rule $c_{ij} \geq \mu^k \Rightarrow x_{ij} = 0$.The obvious choice is therefore to increase the multiplier to its maximum value $\mu^{k+1} = \mu^{\max}$. For many instances of the SSCFLP proposed in the literature, the updating rules above often lead to very small increments in the multiplier.

Therefore a more aggressive updating strategy is taken where the above rules are used and then the multiplier is update according to the rule

$$\mu^{k+1} = \max\{\mu^{k+1}, \mu^k + K\}.$$

That is, we do not allow an increase that is smaller than a constant $K > 0$. This strategy clearly makes the subproblems larger than necessary, but it does also decrease the number of subproblems we need to solve. The above strategies can be summarized by

$$\mu^{k+1} = \begin{cases} \max\{\min_{i\in\mathcal{I},\ j\in\mathcal{J}^0}\{c_{ij}\ :\ c_{ij} > \mu^k\}, \mu^k + K\}, & \text{if } \max\{c_{ij}\ :\ i \in \mathcal{I}, j \in \mathcal{J}^0\} > \mu^k \\ \max\{\min_{i\in\mathcal{I},\ j\in\mathcal{J}}\{c_{ij}\ :\ c_{ij} > \mu^k\}, \mu^k + K\}, & \text{if } \max\{c_{ij}\ :\ i \in \mathcal{I}, j \in \mathcal{J}\} > \mu^k \\ \mu^{\max}, & \text{otherwise} \end{cases}$$

4.3. Cut–and–solve applied to the SSCFLP

The cut–and–solve framework was successfully used for solving the SSCFLP in Gadegaard et al. (2016b) and we therefore use the recommendations given in this paper and use a capacitated facility location problem (CFLP) as a relaxation of the dense problem. Although the CFLP itself is an $\mathcal{NP}$–hard problem, it can be effectively handled by a good MIP solver such as CPLEX or Gurobi. Using the CFLP as a relaxation means that only the x_{ij}–variables are continuously relaxed. This gives a very straightforward way of defining the set of variables Ω defining the piercing cuts. Let (x^{DP}, y^{DP}) be an optimal solution to the CFLP solved as a relaxation of the dense problem, then $\Omega = \{i \in \mathcal{I}\ :\ y_i = 0\}$. The choice of the CFLP as a relaxation is based on the fact that it shares many of the characteristics of the SSCFLP, implying that an optimal solution is often found in one of the first cut–and–solve branching nodes. However, it should be noted that this choice of Ω does not guarantee a feasible solution to each sparse problem. This has implications for the dual ascent algorithm used to solve the sparse problem in the adaptation of Algorithm 5 as we have assumed that the problems solved using the dual ascent algorithm were all feasible. We discuss this shortly in Section 4.6. Furthermore, as the SSCFLP is a binary integer programming problem, we choose to set $\gamma = 0$. In conjunction with the choice of Ω this ensures that all variables y_i and x_{ij}, $i \in \Omega, j \in \mathcal{J}$ are fixed to zero in the sparse problem, reducing these problems considerably.

4.4. Cutting planes for the SSCFLP

It is well established that separating LP solutions from the convex hull of integer solutions to the capacity constraints (9) generally works very well for the SSCFLP (see e.g. Gadegaard et al. (2016b)). However, separating from the convex hull of integer solutions to the capacity constraints alone, means that the assignment constraints (8) are completely ignored. To remedy this, we first reformulate the SSCFLP. To that end, let $\mathcal{I}_1, \mathcal{I}_2 \subseteq \mathcal{I}$ with $\mathcal{I}_1 \neq \emptyset$, $\mathcal{I}_1 \cap \mathcal{I}_2 = \emptyset$, and $\mathcal{I}_1 \cup \mathcal{I}_2 = \mathcal{I}$. Furthermore, introduce new binary variables z_j^l, $l = 1, 2$, defined by

$$z_j^l = \begin{cases} 1, & \text{if } \exists i \in \mathcal{I}_l\ :\ x_{ij} = 1 \\ 0, & \text{otherwise} \end{cases}$$

With these additional binary variables we get the new formulation of the feasible set of the SSCFLP

$$\begin{aligned} P = \big\{(x, y, z) \in \{0,1\}^{n\times m+n+2\times m}\ :\ & \sum_{i\in\mathcal{I}_1} x_{ij} = z_j^1,\ \sum_{i\in\mathcal{I}_2} x_{ij} = z_j^2, \quad \forall j \in \mathcal{J}, \\ & z_j^1 + z_j^2 = 1, \quad \forall j \in \mathcal{J}, \\ & \sum_{j\in\mathcal{J}} d_j x_{ij} \le s_i y_i, \quad \forall i \in \mathcal{I}\big\} \end{aligned}$$

It should be clear, that for any solution $(x, y, z) \in P$ we have that (x, y) is a solution to the SSCFLP. The goal is now to generate valid inequalities for $\mathcal{P} = \text{conv}(P)$ which are violated by an LP solution. To do so, we sum all capacity constraints over $i \in \mathcal{I}_1$ and obtain the valid inequality

$$\sum_{i\in\mathcal{I}_1}\sum_{j\in\mathcal{J}} d_j x_{ij} \le \sum_{i\in\mathcal{I}_1} s_i y_i,$$

which we denote an *effective capacity constraint.* Rearranging terms and using $\sum_{i\in\mathcal{I}} x_{ij} = z^1_j$ yields the inequality

$$\sum_{j\in\mathcal{J}} d_j z^1_j \le \sum_{i\in\mathcal{I}_1} s_i y_i,$$

which can be turned into a standard knapsack constraints by complementing the y–variables, $\gamma_i = 1 - y_i$, thereby yielding:

$$\sum_{j\in\mathcal{J}} d_j z^1_j + \sum_{i\in\mathcal{I}_1} s_i\gamma_i \le \sum_{i\in\mathcal{I}_1} s_i \tag{13}$$

Since (13) is a valid inequality for $\mathcal{P}$, we have that

$$\mathcal{P} \subseteq \operatorname{conv}(\{(z^1,\gamma)\in\{0,1\}^{m+|\mathcal{I}_1|} : \sum_{j\in\mathcal{J}} d_j z^1_j + \sum_{i\in\mathcal{I}_1} s_i\gamma_i \le \sum_{i\in\mathcal{I}_1} s_i\}) =: \mathcal{P}^{z\gamma}(\mathcal{I}_1). \tag{14}$$

This implies that given a solution $(\underline{x},\underline{y})$ to the LP relaxation of the SSCFLP, we can set $\underline{z}^1_j = \sum_{i\in\mathcal{I}_1}\underline{x}_{ij}$ for all $j\in\mathcal{J}$ and $\underline{\gamma}_i = 1-\underline{y}_i$ for all $i\in\mathcal{I}_1$ and then separate $(\underline{z},\underline{\gamma})$ from $\mathcal{P}^{z\gamma}(\mathcal{I}_1)$. Given a hyperplane $\sum_{j\in\mathcal{J}}\pi_j z^1_j + \sum_{i\in\mathcal{I}_1}\psi\gamma_i \le \pi_0$ that separates $(\underline{z}^1,\underline{\gamma})$ from $\mathcal{P}^{z\gamma}(\mathcal{I}_1)$ we can simply reinsert the original variables using the definitions of z^1_j and γ_i and obtain the cutting plane

$$\sum_{j\in\mathcal{J}}\pi_j\big(\sum_{i\in\mathcal{I}_1} x_{ij}\big) \le (\pi_0 - \sum_{i\in\mathcal{I}_1}\psi_i) + \sum_{i\in\mathcal{I}_1}\psi_i y_i.$$

This means that we can use all the classical inequalities such as cover inequalities, lifted cover inequalities, and Fencel inequalities known for the 0-1 knapsack polytope to separate the current LP solution from the polytope $\mathcal{P}^{z\gamma}(\mathcal{I}_1)$.

Note that using exact knapsack separation to separate $(\underline{z}^1,\underline{\gamma})$ from $\mathcal{P}^{z\gamma}(\mathcal{I}_1)$ is equivalent to solving the following separation problem: Find an inequality of the form $\sum_{j\in\mathcal{J}}\pi_j\big(\sum_{i\in\mathcal{I}_1}x_{ij}\big) + \sum_{i\in\mathcal{I}_1}\psi_i y_i \le \pi_0$ that separates $(\underline{x},\underline{y})$ from

$$\operatorname{conv}(\{(x,y)\in\{0,1\}^{n\times m+n} : \sum_{j\in\mathcal{J}}\sum_{i\in\mathcal{I}_1} d_j x_{ij} \le \sum_{i\in\mathcal{I}_2} s_i y_i,\ \sum_{i\in\mathcal{I}_1} x_{ij}\le 1,\ \forall j\in\mathcal{J}\}), \tag{15}$$

or prove that no such inequality exists.

The procedure thus provides a way to exactly separate inequalities of the form

$$\sum_{j\in\mathcal{J}}\pi_j\big(\sum_{i\in\mathcal{I}_1}x_{ij}\big) + \sum_{i\in\mathcal{I}_1}\psi_i y_i \le \pi_0$$

from the polytope (15), which is denoted an *effective capacity polytope with generalized upper bounds.*

The reader might note that for $\mathcal{I}_1 = \mathcal{I}$ and $\mathcal{I}_2 = \emptyset$ we get the well known *total demand* constraint $\sum_{i\in\mathcal{I}} s_i y_i \ge \sum_{j\in\mathcal{J}} d_j$ and for $\mathcal{I}_1 = \{i\}$ we simply get the original capacity constraint $\sum_{j\in\mathcal{J}} d_j x_{ij} \le s_i y_i$. In this paper we consider pairs of facilities, that is $\mathcal{I}_1 = \{i_1, i_2\}$ with $i_1, i_2 \in \mathcal{I}$ and $i_1 \neq i_2$. This yields $n(n-1)/2$ new knapsack structures from which cutting planes can be separated. We have summarized the cutting plane algorithm used to strengthen the program in Algorithm 6.

In order to decrease the computational effort when separating cutting planes we only separate from $\mathcal{P}^{z\gamma}$ if we cannot generate any violated inequalities from the knapsack polytopes defined by single capacity constraints. In Steps 4.2 and 6.2 we follow the recommendations from Kaparis and Letchford (2010)and first attempt to separate a general lifted cover inequality and in case we fail to do so, we resort to Fenchel cutting planes.

```
Input: An instance of the SSCFLP.
Output: A strengthened formulation of the SSCFLP and a vector, $\delta^*$, of optimal dual variables for the assignment constraints.

Step 1: Set $LB^0 = -\infty$ and set the iteration counter $k = 0$.

Step 2: Set $k = k + 1$, solve the linear relaxation of the SSCFLP, and let $(\underline{x}, \underline{y})$ be and optimal solution
        to the LP relaxation with objective function value $LB^k$.

Step 3: If $(\underline{x}, \underline{y})$ is integral, return $(\underline{x}, \underline{y})$ as an optimal solution to the SSCFLP.

Step 4: If $LB^k - LB^{k-1} < \epsilon$, return the strengthened formulation and an optimal dual solution to the
        assignment constraints, $\delta^*$.

Step 5: For each $i \in \mathcal{I}$ do the following:

        Step 5.1: Add all violated implied upper bounds $x_{ij} - y_i \leq 0$ to the formulation.
        Step 5.2: Separate the solution $(\underline{\mathbf{x}}_i)$ from the knapsack polytope $\text{conv}(\{\mathbf{x}_i \in \{0,1\}^{|\mathcal{J}|} : \sum_{j \in \mathcal{J}} d_j x_{ij} \leq s_i\})$ if possible. If a cutting plane, say $\pi^T \mathbf{x}_i \leq \pi_0$, is generated translate
                  it to $\pi^T \mathbf{x}_i \leq \pi_0 y_i$ and add it to the formulation of the SSCFLP.

Step 6: If Step 4 resulted in violated cutting planes, go to Step 6. Else for each pair $i_1, i_2 \in \mathcal{I}$ with
        $i_1 < i_2$ do the following:

        Step 6.1 Create the aggregate solution $\underline{z}_j = \underline{x}_{i_1 j} + \underline{x}_{i_2 j}$ for each $j \in \mathcal{J}$ and $\underline{\gamma}_i = 1 - \underline{y}_i$ for each
                 $i = i_1, i_2$.
        Step 6.2 Separate the point $(\underline{z}, \underline{\gamma})$ from $\mathcal{P}^{z\gamma}(\{i_1, i_2\})$, if possible. If a cutting plane, say $\pi^t z + \psi\gamma \leq$
                 $\pi_0$, is generated, reinsert the original variables and add the cutting plane $\sum_{j \in \mathcal{J}} \pi_j (x_{i_1 j} +$
                 $x_{i_2 j}) \leq (\pi_0 - \psi_1 - \psi_2) + \psi_1 y_{i_1} + \psi_2 y_{i_2}$ to the formulation of the SSCFLP.

Step 7: If $k < K$ and Step 5 or Step 6 resulted in violated cutting planes, go to Step 2, otherwise return
        the strengthened formulation and an optimal dual solution to the assignment constraints, $\delta^*$.
```

Algorithm 6: Summary of the cutting plane algorithm.

4.5. A semi–Lagrangean based dual ascent algorithm using cut–and–solve

An outline of the complete dual ascent algorithm using the cut–and–solve framework for the SSCFLP is given in Algorithm 7. In Step 1 the algorithm is initialized. First, we run the cutting algorithm in order to strengthen the formulation of the SSCFLP and to get a dual solution that can be used to initialize the semi–Lagrangean multiplier. In Steps 2.1 to 2.6 we exploit the fact that the semi–Lagrangean subproblem can be rewritten as a SSCFLP and that we thereby can use a modified version of the cut–and–solve algorithm proposed in Gadegaard et al. (2016b) to solve the semi–Lagrangean subproblem. In Step 2.1 we initialize the upper bound on the subproblem to a large number, and the current best solution of the subproblem is set to the empty solution. In Step 2.2 the dense problem is solved and the solution value is first compared to the best primal feasible solution. As the dense problem provides a lower bound on the optimal solution to the semi–Lagrangean subproblem it is a valid lower bound on the optimal solution, which means that if the value of the dense problem exceeds the value of the current best primal solution, it is optimal. Next, the value of the dense problem is compared to the best solution to the semi–Lagrangean subproblem. If the value of the dense problem exceeds the value of the best solution to the semi–Lagrangean subproblem, the subproblem has been solved to optimality, and we continue to Step 3. If not, we continue to Step 2.3 where the piercing cut is selected. In Step 2.4 the sparse problem is solved. Note that the choice of piercing cuts ensures that the sparse problems always have a feasible solution as the dummy facility has amble capacity

Input: Vectors c and b, matrix A, and feasible set $\mathcal{X}$.
Output: An optimal solution x^* to (1) and optimal multiplier $\mu^* \in \mathcal{M}$.

Step 1: (Initialization) Run Algorithm 6 on the full SSCFLP in order to strengthen the problem and obtain an optimal dual solution to the assignment constraints, say δ_j^*. Next, run a local branching heuristics producing a feasible solution $(\bar{x}, \bar{y})$. If no feasible solution was found, set $UB = \infty$, otherwise set $UB = \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} c_{ij} \bar{x}_{ij} + \sum_{i \in \mathcal{I}} f_i \bar{y}_i$ and $(x^*, y^*) = (\bar{x}, \bar{y})$. Finally, set $k = 0$ and initialize the semi–Lagrangean multiplier to μ^k to suitably small value.

Step 2: (Solving subproblem) Use the cut–and–solve framework to solve the semi–Lagrangean subproblem

- *Step 2.1:* (Initialization of cut–and–solve) Set $sub\_best = \infty$ and initialize the incumbent $(x^k, y^k) = null$.
- *Step 2.2:* (Solution of dense problem) Solve the problem (12) with the integrality requirement on the x_{ij}–variables relaxed. Let Z^{DP} be the optimal solution value and $(x,y)^{DP}$ an optimal solution. If $Z^{DP} \geq UB$, return the incumbent as it is optimal. Else if $Z^{DP} \geq sub\_best$, go to Step 3, with $(x,y)^k$ as an optimal solution to the semi–Lagrangean subproblem. Otherwise, continue to Step 2.3.
- *Step 2.3:* (Piercing cut selection) Let $\Omega = \{i \in \mathcal{I} \ : \ y_i^{DP} = 0\}$.
- *Step 2.4:* (Solution of sparse problem) Solve the problem (12) with all variables $y_i, \ x_{ij}, \ j \in \mathcal{J}, i \in \Omega$ fixed to zero. Let $(x,y)^{SP}$ be an optimal solution to the sparse problem and go to Step 2.5.
- *Step 2.5:* (Incumbent update) If $Z^{SP} \leq sub\_best$, update the subproblem incumbent $(x,y)^k = (x,y)^{SP}$, set $sub\_best = Z^{SP}$, and go to Step 2.6. Else, go to Step 2.7.
- *Step 2.6:* (Feasibility check) If $\sum_{i \in \mathcal{I}} x_{ij}^k = 0$ for some $j \in \mathcal{J}$, go to Step 4. Else, the subproblem solution is primal feasible. If $sub\_best < UB$ update the incumbent $(x^*, y^*) = (x,y)^k$ and set $UB = sub\_best$. Go to Step 2.7.
- *Step 2.7:* (Piercing cut addition) Add the piercing cut $\sum_{i \in \Omega} y_i \geq 1$ to (12) and go to Step 2.2.

Step 3: (Termination check) $(x,y)^k$ is optimal to the subproblem. If $\sum_{i \in \mathcal{I}} x_{ij}^k = 1$ for all $j \in \mathcal{J}$ or $sub\_best \geq UB$, stop with $((x,y)^k, \mu^k)$ as an optimal primal–dual pair. Otherwise, go to Step 4.

Step 4: (Multiplier update) Let $\mathcal{J}^0 = \{j \in \mathcal{J} \ : \ \sum_{i \in \mathcal{I}} x_{ij}^k = 0\}$. Update the multipliers according to the rule

$$\mu^{k+1} = \begin{cases} \max\{\min_{i \in \mathcal{I}, \ j \in \mathcal{J}^0}\{c_{ij} \ : \ c_{ij} > \mu^k\}, \mu^k + K\}, & \text{if } \max_{i \in \mathcal{I}, j \in \mathcal{J}^0}\{c_{ij}\} > \mu^k \\ \max\{\min_{i \in \mathcal{I}, \ j \in \mathcal{J}}\{c_{ij} \ : \ c_{ij} > \mu^k\}, \mu^k + K\}, & \text{if } \max_{i \in \mathcal{I}, j \in \mathcal{J}}\{c_{ij}\} > \mu^k \\ \mu^{\max}, & \text{otherwise,} \end{cases}$$

set $k = k + 1$, and return to *Step 2*.

Algorithm 7: A dual ascent algorithm for the SSCFLP that uses the cut–and–solve framework to solve the semi–Lagrangean subproblems.

to service all customers simultaneously. The solution value of the sparse problem, Z^{SP}, is compared to the best solution to the current subproblem found so far. If the solution to the sparse problem improves the current best solution, the subproblem solution $(x, y)^k$ is updated and the procedure continues to Step 2.6. Otherwise we continue to Step 2.7. If the procedure continues to Step 2.6, it means that the current best solution to subproblem has just been updated. We therefore check if this solution is primal feasible in which case we test if this primal feasible solution improves the incumbent (x^*, y^*). Furthermore, if the current best solution is primal feasible, we continue the cut–and–solve procedure by going to Step 2.7 where the piercing cut is added to the dense problem as we might need to solve the subproblem to optimality. In Step 3 we check if the current best solution is optimal. If the subproblem was solved to optimality, *sub_best* provides a valid lower bound on the optimal solution, so if $sub\_best \geq UB$, we know that (x^*, y^*) is optimal. Conversely, if the subproblem solution is primal feasible, we know, by the construction of the algorithm, that (x^*, y^*) is an optimal solution to the subproblem. By means of point 2 of Theorem 1 we can conclude that $(x, y)^k$ is an optimal solution to the SSCFLP. Otherwise, we go to Step 4 where the semi–Lagrangean multiplier is incremented.

4.6. A cut–and–solve algorithm using semi–Lagrangean based dual ascent for sparse problems

Algorithm 8 outlines the cut–and–solve algorithm for the SSCFLP that utilizes a dual ascent algorithm to solve the sparse problems. In Step 1, the algorithm is initialized by first running the cutting plane algorithm followed by a local branch algorithm. The dense problem is solved in Step 2, and in Step 3 we check if the dense problem produced a solution larger than the incumbent, in which case the incumbent is optimal. In Step 4 we select the set of variables forming the piercing cut. The sparse problem is solved by a dual ascent algorithm in Step 5; first, the iteration counter is set and the semi–Lagrangean dual bound is set to minus infinity in Step 5.1. In Step 5.2 we solve the semi–Lagrangean subproblem. We start by checking if the multiplier is at its upper bound as the subproblem then has to be solved to optimality. If that is not the case, we start by solving the subproblem by a heuristic and if the resulting solution is primal feasible, we solve the subproblem to optimality in Step 5.2.2. In Step 5.3 we perform the termination check of the dual ascent algorithm. If the dual lower bound $\mathcal{L}$ exceeds the value of the incumbent, the sparse problem cannot improve the current best solution and we therefore go to Step 7 where a piercing cut is added. If $\mu^k = \mu^{max}$, we know that the subproblem has been solved to optimality, and if the corresponding solution is not primal feasible, the sparse problem contains no feasible solutions so we continue to Step 7. But if the solution is actually primal feasible, it is also optimal for the sparse problem and we go to Step 6. Otherwise, we need to increase the multiplier in Step 5.4 and the dual ascent algorithm returns to Step 5.2. We update the incumbent in Step 6 and check if the incumbent value falls below the value of the lower bound obtained from the dense problem as the incumbent is then optimal. Finally, in Step 7, the piercing cut is added to the program and we return to Step 2.

5. Computational results

This section describes the results obtained using the algorithms proposed in the previous sections. The primary purpose of the computational study is to test to which extent the integration of the semi–Lagrangean based dual ascent algorithm and the cut–and–solve framework can efficiently solve the single source capacitated facility location problem (SSCFLP). In order for us to test the algorithms we start by identifying the most promising initialization schemes for the semi–Lagrangean dual multiplier. Next, we run the algorithms on three testbeds that have been proposed in the literature and a fourth testbed that consists of large instances generated for this paper. We compare the performance of the two integrated algorithms to the improved cut–and–solve algorithm from Gadegaard et al. (2016b), which is currently one of the fastest algorithms for the SSCFLP.

5.1. Implementation details and test instances

We have tested two different algorithms based on the two frameworks described in Section **??** and compared them to a state–of–the–art solver for the SSCFLP. The first algorithm, denoted `DA-CS`, implements

Input: Vectors c and b, matrix A, and feasible set $\mathcal{X}$.
Output: An optimal solution x^* to (1) and optimal multiplier $\mu^* \in \mathcal{M}$.

Step 1: (Initialization) Run Algorithm 6 on the full SSCFLP in order to strengthen the problem and obtain an optimal dual solution to the assignment constraints, say δ_j^*. Next, run a local branching heuristic producing a feasible solution $(\bar{x}, \bar{y})$. If no feasible solution was found, set *best* $= \infty$, otherwise set $UB = \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} c_{ij}\bar{x}_{ij} + \sum_{i \in \mathcal{I}} f_i \bar{y}_i$ and $(x^*, y^*) = (\bar{x}, \bar{y})$. Finally, set the set of piercing cuts $H = \emptyset$.

Step 2: (Dense problem) Solve the CFLP relaxation of the SSCFLP with all piercing cuts in H added. Let the solution value be Z^{DP} and an optimal solution be $(x, y)^{DP}$.

Step 3: (Termination check) If $Z^{DP} \geq UB$, return the incumbent (x^*, y^*) as it is optimal.

Step 4: (Piercing cut selection) Set $\Omega = \{i \in \mathcal{I} \ : \ y_i^{DP} = 0\}$.

Step 5: (Sparse problem) Use the semi–Lagrangean based dual ascent algorithm to solve the sparse problem.

Step 5.1 (Initialization) Set $k = 0$, $\mathcal{L} = -\infty$, and initialize μ^k to a suitably small value.

Step 5.2 (Solve subproblem) Set $k = k + 1$. If $\mu < \mu^{\max}$, go to Step 5.2.1, else go to Step 5.2.2.

Step 5.2.1 Use a heuristic to obtain a good feasible solution, $(x, y)^k$ to the problem

$$\begin{aligned}
\min \ & \sum_{i \in \mathcal{I}_0} \sum_{j \in \mathcal{J}} c_{ij} x_{ij} + \sum_{i \in \mathcal{I}_0} f_i y_i \\
\text{s.t.:} \ & \sum_{i \in \mathcal{I}_0} = 1, \quad \forall j \in \mathcal{J}, \\
& \sum_{j \in \mathcal{J}} d_j x_{ij} \leq s_i y_i, \quad \forall i \in \mathcal{I}_0, \\
& x_{ij} = y_i = 0, \quad \forall i \in \Omega, j \in \mathcal{J}, \\
& x_{ij} = 0, \quad \forall (i,j) \in \mathcal{I} \times \mathcal{J} : c_{ij} \geq \mu^k, \\
& y_0 = 1, \\
& x_{ij}, y_i \in \{0,1\}, \quad \forall i \in \mathcal{I}, j \in \mathcal{J}.
\end{aligned} \tag{16}$$

If $\sum_{i \in \mathcal{I}} x_{ij} = 0$ for some $j \in \mathcal{J}$, go to Step 5.4, else go to Step 5.2.2.

Step 5.2.2 Solve the problem (16) to optimality. Let $(x, y)^k$ be an optimal solution, let $\mathcal{L}$ be the corresponding solution value, and continue to Step 5.3.

Step 5.3 (Termination check) If $\mathcal{L} \geq UB$, the sparse problem cannot contain improving solutions, hence go to Step 7. If $\mu^k = \mu^{\max}$ and $\sum_{i \in \mathcal{I}} x_{ij}^k = 0$ for some $j \in \mathcal{J}$, the sparse problem is infeasible so go to Step 7. If $\sum_{i \in \mathcal{I}} x_{ij}^k = 1$ for all $j \in \mathcal{J}$, the solution $(x, y)^k$ is optimal for the sparse problem, hence go to Step 6. Otherwise, continue to Step 5.4.

Step 5.4 (Multiplier update) Let $\mathcal{J}^0 = \{j \in \mathcal{J} \ : \ \sum_{i \in \mathcal{I}} x_{ij}^k = 0\}$. Update the multipliers according to the rule

$$\mu^{k+1} = \begin{cases} \max\{\min_{i \in \mathcal{I}, \ j \in \mathcal{J}^0}\{c_{ij} \ : \ c_{ij} > \mu^k\}, \mu^k + K\}, & \text{if } \max_{i \in \mathcal{I}, j \in \mathcal{J}^0}\{c_{ij}\} > \mu^k \\ \max\{\min_{i \in \mathcal{I}, \ j \in \mathcal{J}}\{c_{ij} \ : \ c_{ij} > \mu^k\}, \mu^k + K\}, & \text{if } \max_{i \in \mathcal{I}, j \in \mathcal{J}}\{c_{ij}\} > \mu^k \\ \mu^{\max}, & \text{otherwise,} \end{cases}$$

and return to *Step 5.2.*

Step 6: (Incumbent update) If $\mathcal{L} < UB$, set $UB = \mathcal{L}$ and $(x^*, y^*) = (x, y)^k$. If $Z^{DP} \geq UB$, return (x^*, y^*) as it is optimal.

Step 7: (Adding piercing cut) Add the piercing cut $\sum_{i \in \Omega} y_i \geq \gamma + 1$ to H and go to Step 2.

Algorithm 8: A cut–and–solve algorithm for the SSCFLP that integrates a semi–Lagrangean based dual ascent algorithm to solve the sparse problems.

Algorithm 7 where the cut–and–solve framework is used to solve the semi–Lagrangean subproblems both heuristically and exactly. Algorithm `CS-CPX` is the cut–and–solve algorithm described in Algorithm 3. We have used the implementation proposed in Gadegaard et al. (2016b) as this algorithm was proven to be effective for solving the SSCFLP. It serves mainly as a benchmark algorithm as nothing new is proposed here. The algorithm is enhanced with the cutting planes proposed in Section 4.4 to make the comparison fair. Finally, `CS-DA` augments the implementation of the cut–and–solve algorithm proposed in Gadegaard et al. (2016b) such that each sparse problem is solved using a semi–Lagrangean based dual ascent algorithm. It uses a standard implementation of Algorithm 2 to solve these problems. Each semi–Lagrangean subproblem arising when solving the sparse problems of the cut–and–solve algorithm is solved using CPLEX which has been strengthened by a cut callback employing the cutting plane algorithm Algorithm 6. We use CPLEX as a heuristic by setting the relative optimality gap to a small positive value. When a solution of sufficient quality has been found, the best solution is checked for primal feasibility. If the solution is primal feasible, the subproblem is solved to optimality. If, on the other hand, the solution is infeasible, the dual multiplier is increased and a new subproblem is solved.

The cutting plane algorithm, Algorithm 6, uses general lifted cover inequalities (GLCI) to separate fractional points from each individual capacity constraint. If no violated GLCI can be generated, we use the separation algorithm for Fenchel inequalities developed in Gadegaard et al. (2016b). Finally, if no cuts could be separated from any of the capacity constraints, we attempt to separate the aggregate solution from the polytope $\mathcal{P}^{z\gamma}$ defined in (14). As in the case of the individual capacity constraints, we first attempt to separate using GLCIs and only if that fails, do we use the exact knapsack separation algorithm.

In all algorithms we initialize the upper bound by running a local branching heuristic on the full problem. The heuristic is similar to the one described in Gadegaard et al. (2016b); first, we run a local branching heuristic on the y–variables in order to get a good initial solution. The second phase then takes the best solution from the first phase and improves it, again by using a local branching heuristic. After each phase we use a fast two–opt local search that swaps customers between open facilities to find a local optimum given the current set of open facilities.

We carried out preliminary experiments using a version of Algorithm 1 in which we used the solver developed in Gadegaard et al. (2016b) as a subproblem solver. It turned out that this methodology was very inferior to `DA-CS`, `CS-DA`, and `CS-CPX`, even for the smallest instances. Therefore, we do not report on the results obtained using this methodology.

All algorithms have been coded in C and C++ and compiled using gcc and g++ with optimization option O3 and C++11 features enabled. As solver for the MIPs arising in the subproblems, we have used CPLEX 12.6 with callbacks. The ParallelMode switch is set to deterministic such that different runs can be compared. All codes are publicly available (see Gadegaard, Klose, and Nielsen (2016a)).

The tests have been performed on instances from four different testbeds; the first testbed, TB1, consists of 57 instances used in Díaz and Fernández (2002) ranging in size from instances with 10 facilities and 20 customers to instances with 30 facilities and 90 customers. The second testbed, TB2, contains instances having 10 to 30 facilities and 50 to 200 customers and has been used in Holmberg et al. (1999). Yang et al. (2012) proposed a testbed consisting of relatively large instances ranging in size from 30 facilities and 200 customers to 80 facilities and 400 customers, TB3. These instances range in size from 30 facilities and 200 customers to 80 facilities and 400 customers. The last testbed, TB4, consists of 30 new instances generated according to the procedure proposed by Cornuejols, Sridharan, and Thizy (1991). First, the demand at customer j is generated from $U[5, 35]$, where $U[a, b]$ denotes a uniform distribution on the interval $[a, b]$. Next the positions of the customers and facilities are generated as random points in a unit square. The assignment costs are then set equal to $c_{ij} = d_j\delta_{ij}$, where δ_{ij} denotes the Euclidean distance between customer j and facility i multiplied by 10. Next, the capacity s_i at facility i is generated uniformly from $[10, 160]$. We then scale the capacities such that the ratio $\sum_{i\in\mathcal{I}} s_i/\sum_{j\in\mathcal{J}} d_j = R$ where $R \in \{3, 5, 7\}$. Finally, the fixed operating costs are generated as $f_i = U[0, 90] + U[100, 110]\sqrt{s_i}$. We note that all parameters have been rounded to the nearest integer such that we avoid numerical instability. Two groups of instances were generated: the first group contains 15 instances of size 80×500 and the second group consists of 15 instances of size 100×400. Each group consists of 5 instances having a capacity–to–demand ratio of $R = 3$, $R = 5$, and $R = 7$, respectively. Table 1 provides an overview of the table headings.

Table 1: Column headers of the tables in Section 5.

Header	Description
#	Number of instances over which the average is taken.
size	Size of the instance given by $\lvert\mathcal{I}\rvert \times \lvert\mathcal{J}\rvert$.
R	Ratio between total capacity and total demand, that is $R = \sum_{i\in\mathcal{I}} s_i / \sum_{j\in\mathcal{J}} d_j$.
it	Number of iterations used by the dual ascent algorithm before optimality was proven.
it^{opt}	Number of iterations in which the subproblem was solved to optimality.
% x–left	The percentage of the x_{ij}–left in the final subproblem of the dual ascent algorithm.
CPU	The total time used by the algorithm in question measured in seconds.

5.2. Testing the initialization of the dual multiplier

In this section we test three initialization schemes for the semi–Lagrangean multiplier. We run the two algorithms `DA-CS` and `CS-DA` on the smaller instances from TB1 and TB2. Each algorithm is run with the following three initialization schemes:

Dual: First, we apply the cutting plane algorithm separating from the knapsack structures. When no more cutting planes can be found, we set $\mu^0 = \max_{j\in\mathcal{J}}\{\delta^*_j\}$, where δ^* is a vector of optimal dual multipliers for the assignment constraints.

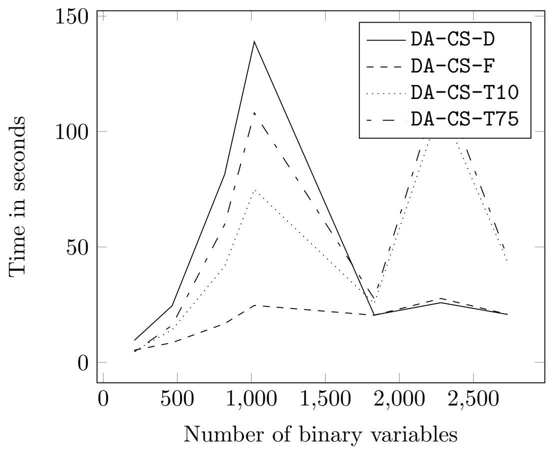
Threshold: We set a minimum number of assignments that should be allowed for each customer, say T. Then $\mu^0 = \min\{n \in \mathbb{N} \,:\, |\{i \in \mathcal{I} \,:\, c_{ij} \leq n\}| \geq T, \forall j \in \mathcal{J}\}$.

Feasible Let $(\bar{x}, \bar{y})$ be a good feasible solution, then $\mu^0 = \min\{c_{ij} \,:\, c_{ij} > c(\bar{x}), i \in I, j \in J\}$, where $c(\bar{x}) = \max\{c_{ij} \,:\, \bar{x}_{ij} > 0\}$.

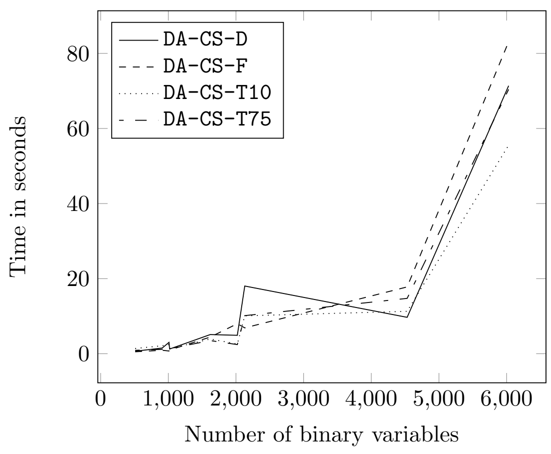
We append a `D`, `T`, or `F` to the name of the algorithm to denote that initialization strategy Dual, Threshold, or Feasible is used. Furthermore, we test both an aggressive and a more conservative value of T for the `T`–option; for the aggressive option we let $T = \lceil|\mathcal{I}|0.75\rceil$, thereby allowing for relatively large subproblems and consequently fewer iterations of the dual ascent algorithm. For the more conservative value, we let $T = \lceil|\mathcal{I}|0.1\rceil$, effectively eliminating about 90 percent of the x_{ij}–variables in the first subproblem. This strategy might lead to more, but smaller subproblems solved. We denote the aggressive and the conservative versions of the threshold strategy `T75` and `T10`, respectively. Thus, we get a total of 8 different algorithms, namely `DA-CS-i` and `CS-DA-i`, for `i`$\in$ `{D,F,T10,T75}`.

Note that we do not have dual information available for the algorithm `CS-DA` as the cutting plane algorithm used for this algorithm is run at the root node of the cut–and–solve tree, and the dual ascent algorithm is run at each sparse problem. This means that we use the optimal dual vector found at the root node of the cut–and–solve tree to initialize the semi–Lagrangean multiplier when the initialization strategy `D` is specified.

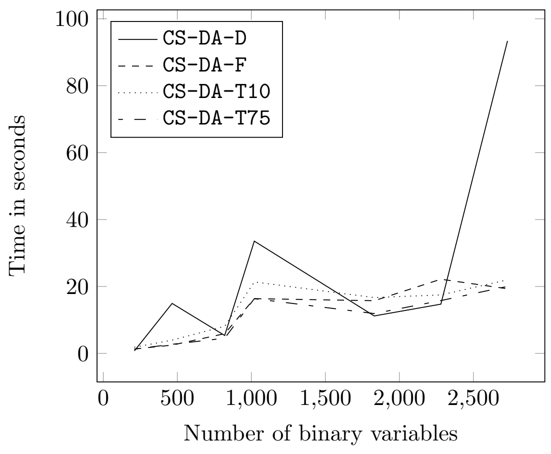
Figure 2 summarizes the performance of the 2 different algorithms and the four initialization schemes. For the `DA-CS` algorithm, the initialization strategy that seems to be the overall best and the most stable is `F`. The results obtained on TB2 are very consistent over all four initialization strategies, but in TB1 all strategies become very unstable except for `F`. As regards the `CS-DA` algorithm, however, we get the best performance using option `T75`, initializing the multiplier using the aggressive threshold value. In Table 2, we have summarized the number of dual ascent iterations and the percentage of the variables left in the last iteration of the dual ascent algorithms, that is $|\{(i,j) \in \mathcal{I}\times\mathcal{J} \,:\, c_{ij} < \mu^*\}|/(|\mathcal{I}|\cdot|\mathcal{J}|)$. For the algorithm `CS-DA`, these numbers are calculated as an average over all sparse problems solved in the cut–and–solve algorithm for the particular instance. For `DA-CS` the `D` initialization strategy results in few iterations of the algorithm and at the same time, the final subproblem remains small. For the `CS-DA`, Table 2 seems to support the observations from Figure 2, namely that the initialization strategy `T75` works the best. Therefore, based on Figure 2 and Table 2, the tests on the larger instances of TB3 and TB4 will be carried out with the initialization strategies `D` and `F` for the `DA-CS` algorithm, whereas `CS-DA` will only be initialized using the `T75` strategy.



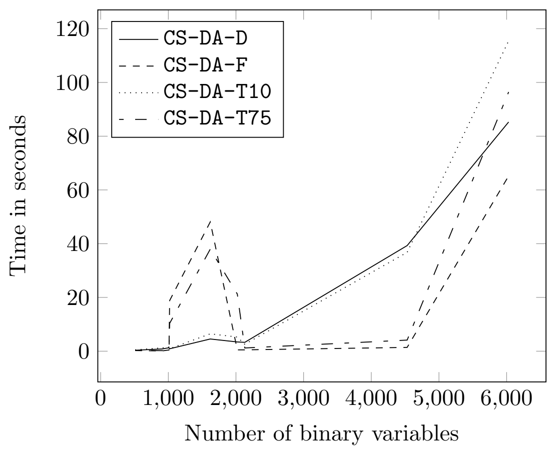
(a) Results of the three initialization strategies obtained from TB1 using `DA-CS`.



(b) Results of the three initialization strategies obtained from TB2 using `DA-CS`.



(c) Results of the three initialization strategies obtained from TB1 using `CS-DA`.



(d) Results of the three initialization strategies obtained from TB2 using `CS-DA`.

Figure 2: Overview of the performance of the four different initialization strategies. Figures 2a and 2b show results for `DA-CS` whereas Figures 2c and 2d show the results obtained with `CS-DA`.

Table 2: Overview of the average number of dual ascent iterations and the average number of x_{ij}–variables left in the last iteration of the dual ascent algorithms.

	`DA-CS`				`CS-DA`			
	`D`	`F`	`T10`	`T75`	`D`	`F`	`T10`	`T75`
Dual ascent iterations	1.1	4.7	5.42	1.52	5.7	4.8	9.9	2.5
Percentage of variables left	67.1	65.2	63.1	95.8	32.9	50.0	35.3	43.5

Table 3: Results obtained for TB3 using the semi–Lagrangean based dual ascent algorithms using cut–and–solve to solve the semi–Lagrangean subproblems, `DA-CS`.

				DA-CS-F				DA-CS-D			
TB	#	size	R	it	it^{opt}	$\%x$-left	CPU	it	it^{opt}	x_{ij}–left	CPU
3	5	30×200	$1.73 - 1.98$	4.6	1.0	81.2	171.38	1.0	1.0	99.2	100.25
	5	30×300	$2.88 - 3.49$	7.6	1.0	79.0	677.47	1.0	1.0	98.1	256.34
	5	60×300	$3.42 - 5.78$	5.4	1.0	73.0	1627.22	1.0	1.0	94.1	364.66
	5	80×400	$3.50 - 7.30$	5.4	1.0	78.9	5035.14	1.0	1.0	92.4	1712.35
Averages				5.8	1.0	78.0	1877.80	1.0	1.0	95.9	608.40
4	5	80×500	3.00[1]	3.0	1.0	67.9	183.15	1.0	1.0	75.5	2353.94
	5		5.00	1.8	1.0	77.7	1751.20	1.0	1.0	69.4	1760.34
	5		7.00	1.8	1.0	85.4	4861.61	1.6	1.0	71.1	4346.43
	5	100×400	3.00	3.8	1.0	69.8	1455.07	1.0	1.0	84.7	573.26
	5		5.00	2.4	1.0	76.1	635.93	1.2	1.0	75.3	577.80
	5		7.00	2.0	1.0	79.5	1499.78	1.0	1.0	70.1	1547.11
Averages				2.4	1.0	76.6	1796.27	1.1	1.0	74.1	1888.11

[1] We were not able to solve instance $p4$ of size 80×500. The program was killed by the operating system as the size of the branching tree grew too large.

5.3. *Analysis of the `DA-CS` algorithms*

In this section we analyze the results obtained with the algorithms `DA-CS-F` and `DA-CS-D`. Table 3 shows the results obtained with `DA-CS-F` and `DA-CS-D` for the instances of testbeds TB3 and TB4. We have chosen not to reproduce the optimal solution values and have instead aggregated the results by taking the average over five instances having the same size (for detailed information about the instances of TB3 consult Yang et al. (2012) and for information on the new instances of TB4 see Appendix A).

Table 3 shows that for the instances of TB3, `DA-CS-D` clearly outperforms `DA-CS-F`, as the latter uses more than three times as much CPU time on average compared to `CS-DA-D`. Even though the final subproblem solved by `DA-CS-F` is reduced to 78 percent of the original problem on average, this is not enough to counterbalance the additional iterations needed when the initialization strategy `F` is used compared to `D`. It is worth noting that the initialization strategy `D` provides an optimal dual multiplier for all instances of TB3, leading to only one subproblem that needs to be solved. In TB4 the performance of the two initialization strategies seem more equal as the CPU times used by the two algorithms are almost identical and differ by less than 5 percent. In TB4 the initialization strategy `D` is again very good at predicting an optimal dual multiplier as only 1.1 iterations are needed by the dual ascent algorithm on average. A very interesting point is that the modified cut–and–solve algorithm seems to work very well as a subproblem solver as it needed to solve only one subproblem to optimality in all instances of TB3 and TB4. This suggests that the truncated cut–and–solve algorithm is indeed a very effective heuristic for the SSCFLP.

5.4. *Analysis of the `CS-DA` algorithm*

Table 4 summarizes the results obtained with the cut–and–solve algorithm using the dual ascent algorithm for solving the sparse problems. It is evident that the cut–and–solve framework in conjunction with the semi–Lagrangean based dual ascent algorithm is very effective in reducing the size of the integer linear programs that need to be solved. In the column headed “SP size” we report the average size of the final sparse problem as a percentage of the original instance size. On average, the sparse problems consist of less than 10 percent of the variables in the original problem. We note that the results confirm the findings from Gadegaard et al. (2016b), that is that the CFLP relaxation of the SSCFLP provides a good approximation as regards the facilities to be opened, and we note that the cut-and-solve algorithm requires less than 3 iterations to find an optimal solution. In addition, the dual ascent algorithm used to solve the subproblems only uses about 2 iterations on average to solve the sparse problems to optimality. Furthermore, we observed that CPLEX very quickly found near optimal solutions to the semi–Lagrangean subproblems and that the main effort in the subproblems was spent on increasing the lower bound. Memory consumption therefore

Table 4: Results obtained on TB3 and TB4 using cut–and–solve algorithm incorporating a dual ascent algorithm for solving the sparse problems, `CS-DA`.

				Iterations			
TB	#	size	R	CS[1]	DA[2]	SP size	CPU
3	5	30×200	$1.73 - 1.98$	2.4	2.2	15.5	75.55
	5	30×300	$2.88 - 3.49$	3.8	1.7	5.1	1115.67
	5	60×300	$3.42 - 5.78$	2.0	2.4	8.9	80.32
	5	80×400	$3.50 - 7.30$	3.0	1.7	4.8	337.31
Averages				2.8	2.0	8.5	402.21
4	5	80×500[3]	3	2.0	2.0	9.0	2347.01
	5		5	2.3	1.6	7.0	2406.79
	5		7	2.0	1.8	7.1	7157.59
	5	100×400[3]	3	2.5	1.0	8.4	709.42
	5		5	2.6	1.6	5.9	659.62
	5		7	2.4	2.2	5.2	1735.84
Averages				2.3	1.65	7.3	2398.84

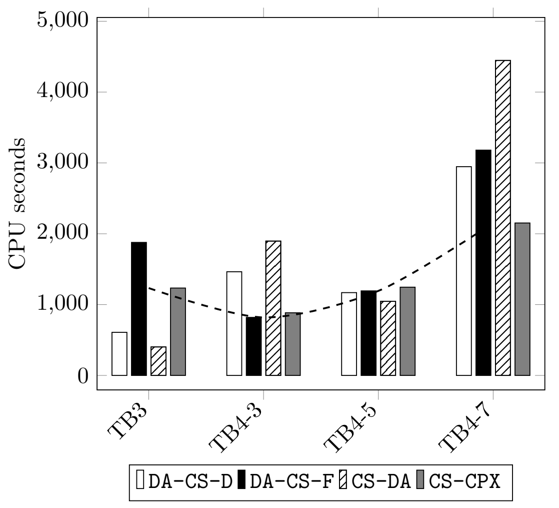
[1] Number of iterations of the cut–and–solve algorithm.
[2] Average number of iterations of the dual ascent algorithm used to solve a sparse problem.
[3] We were not able to solve instance $p4$ of size 80×500. The program was killed by the operating system as the size of the branching tree grew too large.

only became prohibitive in one instance. In Section 5.5, we discuss how the solver proposed in Gadegaard et al. (2016b) was immensely challenged by the task of finding a feasible solution to the sparse problems when the ratio between total capacity and total demand was small. The `CS-DA` algorithm performs slightly better on the instances of TB3 compared to the `DA-CS` algorithms analyzed in the previous section. However, for the larger instances of TB4, the roles are reversed. One reason for this change in performance is that the `CS-DA` solves a CFLP as a relaxation of the full problem. When the instances get larger, this large CFLP becomes more time consuming to solve. The CFLP solved as a relaxation of the semi–Lagrangean subproblem in `DA-CS` is, however, significantly reduced by the rule $c_{ij} \geq \mu \Rightarrow x_{ij} = 0$.

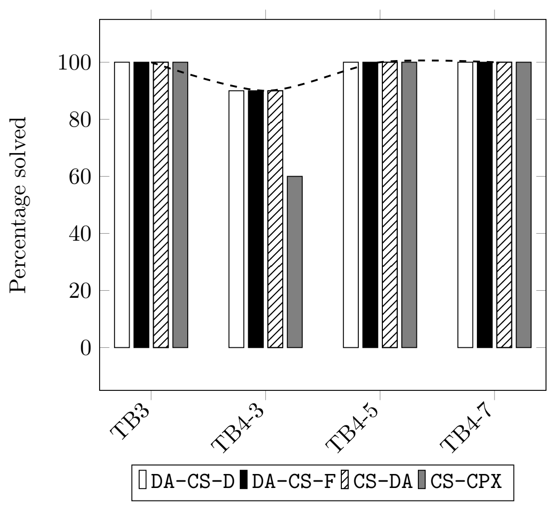
5.5. *Comparing with a state–of–the–art solver for the SSCFLP*

Figure 3 depicts the performance of the three algorithms `DA-CS`, `CS-DA`, and `CS-CPX`. From Figure 3a we see that none of the algorithms perform consistently better than the others. For the instances of TB3, `CS-DA` performs extremely well, but this performance deteriorates as the instances and the capacity to demand ratios increase. We also see that the augmented cut–and–solve algorithm, `CS-CPX`, performs very well on most of the instances, which confirms that this algorithm is indeed very efficient for solving the SSCFLP. Figure 3a shows that the `CS-CPX` algorithm performs just as well or even better than the semi–Lagrangean based algorithms. It should be noted that the time–averages are taken over instances that were actually solved. This makes the `CS-CPX` look better than is actually the case. When we look at Figure 3b we see that the dual ascent based algorithms are more robust compared to `CS-CPX` in case the instances have a small ratio between total capacity and total demand. `CS-CPX` failed on four of the 10 instances of TB4 having a ratio $R = 3$. In all four cases CPLEX failed to find a feasible solution to the first sparse problem and consequently no nodes could be fathomed based on bound. The operating system would relatively quickly kill the program due to the excessive memory consumption.

The results we have obtained suggests that a hybrid solver, which based on the characteristics of the instance chooses the algorithmic approach that should be taken. If the instance has a large R–value or if it is small, then the cut–and–solve algorithm `CS-CPX` should be used. On the other hand, if the instance



(a) Time used by the four algorithms on the large instances.



(b) Number of instances solved by the algorithms.

Figure 3: Comparison with the cut–and–solve algorithm proposed in Gadegaard et al. (2016b). The i in TB4-i indicates the ratio between total demand and total capacity.

is large and has a small R–value the solver should use the dual ascent algorithm `DA-CS-F`. The dashed line in Figure 3 depicts the performance of such an approach, showing that in most cases this would be a very efficient solution procedure for the SSCFLP.

6. Conclusion

In this paper we have proposed two new ways of integrating semi–Lagrangean based dual ascent and cut–and–solve for the single source capacitated facility location problem (SSCFLP). The first approach used a cut–and–solve algorithm as a subproblem solver in a dual ascent algorithm where the cut–and–solve algorithm only solved the subproblems to optimality if a good primal feasible solution was found. The second approach augmented an existing cut–and–solve algorithm by solving the sparse problems using a dual ascent algorithm based on a semi–Lagrangean relaxation of the SSCFLP. We proposed to update the semi–Lagrangean multiplier based on the solution to the semi–Lagrangean subproblem and tested four distinct initialization strategies and found that for the cut–and–solve algorithm using dual ascent for solving the subproblems, the initialization strategy that worked the best was setting the initial value of the dual multiplier such that a 25 percent reduction of the subproblem was obtained. The results were slightly more ambiguous for the dual ascent algorithm using cut–and–solve as the subproblem solver. Here, initializing using a feasible solution or using the dual multipliers of the assignment constraints worked equally well. In order to speed up the computations and limit the memory consumption we proposed a new way of generating cutting planes for a substructure of the SSCFLP combining the capacity constraints and the assignment constraints.

We showed empirically that the dual ascent based algorithms solved more instances compared to an efficient cut–and–solve algorithm proposed in the literature, when the instances were large and had a tight capacity to demand ratio. We argued that the primary reason is that the feasibility problem corresponding to the Lagrangean subproblem is polynomially solvable in contrast to the $\mathcal{NP}$–hard feasibility problem of the SSCFLP. With these new methodologies it was possible to increase the size of the instances solvable by almost 25 percent.

In the future, it will be interesting to investigate whether improved updating strategies for the dual multiplier and fixation of variables based on Lagrangean penalty tests can lead to improved performance through fewer subproblems. It is also of interest to examine if the methodology can be used to solve more complex location problems such as dynamic or multi-stage models.

7. Acknowledgments

The author would like to thank Associate Professors Andreas Klose and Lars Relund Nielsen for insightful comments and suggestions. This work was partially supported by a grant from Købmand Ferdinand Sallings Mindefond.

Appendices

A. Detailed information on the new instances

Table A.5 gathers information on the new instances generated for this paper. The columns headed *id*, *size*, and R contain the name of the instance, the size of the instance ($|\mathcal{I}| \times |\mathcal{J}|$), and the ratio between total supply and total demand, respectively. The next three columns headed LB^{LP}, LB^{CP}, and Gap closed report the lower bound obtained from the LP relaxation before adding cuts, the lower bound produced by the cutting plane algorithm, Algorithm 6, and the percentage gap closed by the cutting planes. The last two columns headed UB and $\%gap$ contain the value of the best known upper bound and the percentage gap between this upper bound and the best known lower bound. All instances are publicly available (see Gadegaard, Klose, and Nielsen (2016c)).

References

Avella, P., Boccia, M., Sforza, A., Vasilév, I., 2008. An effective heuristic for large-scale capacitated facility location problems. Journal of Heuristics 15 (6), 597–615.

Barceló, J., Casanovas, J., 1984. A heuristic Lagrangean algorithm for the capacitated plant location problem. European Journal of Operational Research 15 (2), 212–226.

Barceló, J., Fernández, E., Jörnsten, K., 1991. Computational results from a new lagrangean relaxation algorithm for the capacitated plant location problem. European Journal of Operational Research 53 (1), 38–45.

Barceló, J., Hallefjord, Å., Fernández, E., Jörnsten, K., 1990. Lagrangean relaxation and constraint generation procedures for capacitated plant location problems with single sourcing. Operations-Research-Spektrum 12 (2), 79–88.

Beasley, J., 1993. Lagrangean heuristics for location problems. European Journal of Operational Research 65 (3), 383 – 399.

Beltran, C., Tadonki, C., Vial, J., 2006. Solving the p-Median Problem with a Semi–Lagrangian Relaxation. Computational Optimization and Applications 35 (2), 239–260.

Beltran-Royo, C., Vial, J., Alonso-Ayuso, A., 2012. Semi–Lagrangian relaxation applied to the uncapacitated facility location problem. Computational Optimization and Applications 51, 287–409.

Bitran, G., Chandru, V., Sempolinski, D., Shapiro, J., 1981. Inverse optimization: An application to the capacitated plant location problem. Management Science 27 (10), 1120 – 1141.

Chen, C., Ting, C., 2008. Combining Lagrangian heuristic and ant colony system to solve the single source capacitated facility location problem. Transportation Research Part E: Logistics and Transportation Review 44 (6), 1099 – 1122.

Climer, S., Zhang, W., 2006. Cut-and-solve: An iterative search strategy for combinatorial optimization problems. Artificial Intelligence 170 (8), 714–738.

Cornuejols, G., Sridharan, R., Thizy, J., 1991. A comparison of heuristics and relaxations for the capacitated plant location problem. European Journal of Operational Research 50 (3), 280–297.

Cortinhal, M., Captivo, M., 2003. Upper and lower bounds for the single source capacitated location problem. European Journal of Operational Research 151 (2), 333 – 351.

Díaz, J., Fernández, E., 2002. A branch-and-price algorithm for the single source capacitated plant location problem. Journal of the Operational Research Society 53, 728–740.

Gadegaard, S., Klose, A., Nielsen, L., 2016a. Implementations of a cut–and–solve algorithm for the single source capacitated facility location problem. `https://github.com/SuneGadegaard/SSCLPsolver`, source code (v1.2.0).

Gadegaard, S., Klose, A., Nielsen, L., 2016b. An improved cut–and–solve algorithm for the single source capacitated facility location problem, submitted.

Gadegaard, S., Klose, A., Nielsen, L., 2016c. Instances for facility location problems. `https://github.com/SuneGadegaard/Instances`, source code.

Guastaroba, G., Speranza, M. G., 2012. Kernel search for the capacitated facility location problem. Journal of Heuristics 18 (6), 877–917.

Holmberg, K., Rönnqvist, M., Yuan, D., 1999. An exact algorithm for the capacitated facility location problems with single sourcing. European Journal of Operational Research 113 (3), 544–559.

Table A.5: Detailed information on the new instances generated for this paper.

id	size	R	LB^{LP}	LB^{CP}	%Gap closed	UB	%gap
p1	80×500	3	29194.7	31341.5	90.34	31571	0
p2			29291.0	31352.4	91.17	31552	0
p3			30417.0	32714.5	93.93	32863	0
p4			30087.6	32179.3	48.68	34384	7
p5			28405.2	30781.1	92.92	30962	0
p6		5	20183.6	24957.7	96.36	25138	0
p7			20038.5	24927.1	96.41	25109	0
p8			20998.6	25836.9	96.33	26021	0
p9			20537.3	24829.1	95.63	25025	0
p10			19784.6	24830.4	98.39	24913	0
p11		7	16214.0	22981.4	97.58	23149	0
p12			15792.8	22501.6	98.77	22585	0
p13			16847.2	23936.1	97.75	24099	0
p14			16395.2	22928.8	97.87	23071	0
p15			15948.1	23246.1	97.54	23430	0
p16	100×400	3	34992.1	36278	87.60	36460	0
p17			32707.5	33933.4	83.82	34170	0
p18			32533.1	33909.8	88.94	34081	0
p19			30153.2	31584.7	86.56	31807	0
p20			32463.6	33570.8	86.20	33748	0
p21		5	22870.7	25783.2	95.76	25912	0
p22			21720.8	24936	96.49	25053	0
p23			21782.7	25268	95.35	25438	0
p24			20207.6	23699.9	94.84	23890	0
p25			21578.2	24845.9	94.09	25051	0
p26		7	17609.2	22219.7	98.48	22291	0
p27			16823.0	21765.6	96.44	21948	0
p28			17073.3	22234.1	97.09	22389	0
p29			15835.8	21059.9	98.23	21154	0
p30			16771.3	21696.5	95.71	21917	0

Jörnsten, K., Klose, A., 2015. An improved lagrangian relaxation and dual ascent approach to facility location problems. Computational Management Science, 1–32.

Kaparis, K., Letchford, A., 2010. Separation algorithms for 0-1 knapsack polytopes. Mathematical Programming 124 (1-2), 69–91.

Klincewicz, J., Luss, H., 1986. A Lagrangian relaxation heuristic for capacitated facility location with single-source constraints. Journal of the Operational Research Society 37, 495–500.

Monabbati, E., 2014. An application of a Lagrangian-type relaxation for the uncapacitated facility location problem. Japan Journal of Industrial and Applied Mathematics 31 (3), 483–499.

Neebe, A., Rao, M., 1983. An algorithm for the fixed-charge assigning users to sources problem. Journal of the Operational Research Society 34, 1107–1113.

Pirkul, H., 1987. Efficient algorithms for the capacitated concentrator location problem. Computers & Operations Research 14 (3), 197 – 208.

Pisinger, D., Sep. 1997. A Minimal Algorithm for the 0-1 Knapsack Problem. Operations Research 45 (5), 758–767.

Sridharan, R., 1993. A Lagrangian heuristic for the capacitated plant location problem with single source constraints. European Journal of Operational Research 66 (3), 305 – 312.

Yang, Z., Chu, F., Chen, H., 2012. A cut-and-solve based algorithm for the single-source capacitated facility location problem. European Journal of Operational Research 221 (3), 521–532.