UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

BERNARDO SULZBACH

The single-source capacitated multi-source Weber problem with fixed opening costs and distance limits

Work presented in partial fulfillment
of the requirements for the degree of
Bachelor in Computer Science

Advisor: Prof. Dr. Marcus Ritt

Porto Alegre
2019

ABSTRACT

The single-source capacitated multi-source Weber problem with fixed costs and distance limits (SSC-MSWP-FC-DL) is an NP-hard problem. The aim is to open facilities to satisfy the demand of a set of customers. This involves determining the number of facilities to open, where to open them, and to which (single) facility each customer is allocated. This allocation has to take into consideration the maximum capacity of the facilities and the distance limit of their service. In contrast to discrete facility location problems, there are no candidate positions for the facilities: they can be opened at any position in the Euclidean plane. The objective is to minimize the total cost, that is, the sum of the Euclidean distance between each customer and its assigned facility multiplied by the customer demand and the cost of opening the facilities. A continuous location-allocation method which does not violate the constraints is proposed and tested on 75 instances from the literature and 25 instances derived from base instances from the literature. The proposed method produces competitive results to those published on our problem without the single-source capacity constraint and finds solutions on average 37% better than a commercial solver in 0.03% of the time the solver was given.

Keywords: Continuous location-allocation. Weber problem. Multiple sources. Capacity limits. Distance limits.

O problema de Weber de múltiplas fontes com limite de capacidade de fonte única com custos fixos de abertura e limites de distância

RESUMO

O problema de Weber de múltiplas fontes com limite de capacidade de fonte única com custos fixos e limites de distância é um problema NP-difícil. O objetivo é abrir instalações para satisfazer a demanda de um conjunto de clientes. Isso envolve determinar o número de instalações a serem abertas, onde abri-las e para qual (única) instalação cada cliente será alocado. Essa alocação precisa levar em conta a capacidade máxima das instalações e o limite de distância do serviço. Diferentemente dos problemas discretos de localização de instalações, não existem posições candidatas às instalações: elas podem ser abertas em qualquer posição do plano Euclidiano. O objetivo é minimizar o custo total, isto é, a soma da distância Euclidiana entre cada cliente e a instalação a que ele está alocado multiplicada pela demanda do cliente e o custo de se abrir as instalações. Um método de localização-alocação contínua de que não viola as restrições é proposto e testado em 75 instâncias da literatura e 25 instâncias obtidas a partir de instâncias existentes na literatura. O método proposto produz resultados competitivos aos publicados sobre o nosso problema sem a restrição de capacidade de fonte única e encontra soluções em média 37% melhores do que as encontradas por um solver comercial em 0,03% do tempo dado para o solver.

Palavras-chave: Localização-alocação contínua. Problema de Weber. Múltiplas fontes. Limites de capacidade. Limites de distância.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

SSC	Single-Source Capacitated
WP	Weber Problem
MSWP	Multi-Source Weber Problem
FC	Fixed Cost
DL	Distance Limit
FLP	Facility Location Problem
GAP	Generalized Assignment Problem
SOCP	Second-Order Cone Program
MISOCP	Mixed-Integer Second-Order Cone Program

CONTENTS

1 INTRODUCTION

In the Weber problem (WP), the goal is to locate a source on a plane given a set of n customer positions and demands. The objective is to minimize the sum of the transportation costs (the Euclidean distance to a customer multiplied by the demand of this customer). According to Beck and Sabach (2015), the Weber problem has also been called the Fermat-Weber problem, the Fermat-Torricelli problem, and the Steiner problem. However, the Weber problem is the most frequently used name for the problem nowadays. The multi-source Weber problem (MSWP) is a natural extension of the single-source Weber problem whose objective is to locate m new facilities instead of a single one. Its objective is to minimize the overall transportation cost that is given by the weighted sum of the distances between each destination and its assigned source. Examples of the WP and the MSWP can be seen in Figure 1.1.

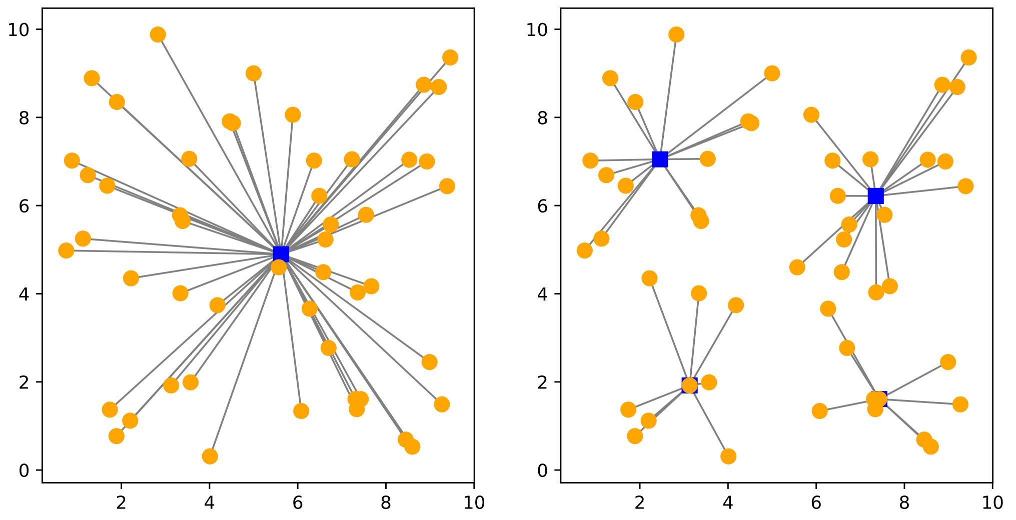
Classically, the only constraint is that each customer must have all of its demand satisfied by the sources. To the best of our knowledge, tackling the MSWP with several additional constraints at once has not yet been done in the literature. Motivated by the necessities of real-world systems, three additional constraints seem to be the most relevant: the single-source capacity limit (SSC), the fixed opening cost (FC) and the coverage distance limit (DL).

The capacity constraint is frequently present when dealing with utility distribution. For instance, a water pump has a flow rate limit and cellphone service networks have a capacity limit. The single-source capacity constraint (which implies that each customer may be serviced by only one source) is justified by the fact that often consuming a resource from multiple sources may require higher adhesion fees for customers or be infeasible due to the nature of the resource.

The fixed opening cost is considered a valuable property as the decision of how many sources to open is not an obvious one, even more so when capacity and distance limits are introduced. Therefore, this decision is better delegated to the solver.

The coverage distance limit of the facilities arises from applications where the quality of the service decays with the travelled distance. For instance, it can be associated with the voltage drop in energy systems due to the resistance of the cables (KOCAMAN; HUH; MODI, 2012) or the pressure loss in water systems due to the friction in the pipes (DOUGLAS; MATTHEWS, 1996 apud KOCAMAN; HUH; MODI, 2012). Both of these limitations are linearly increasing with distance, but other relations are possible (e.g.

Figure 1.1: Solutions for the WP (left) and the MSWP with $m = 4$ (right) for $n = 50$ clients. Clients are orange, sources are blue.



electromagnetic signal strength usually follows an $1/p^2$ law).

1.1 Notation

In this thesis, we consider several variants of Weber problems, which have long, sometimes inconsistent names and abbreviations in the literature. To simplify and ease comprehensions we propose a new, unified notation for Weber problems. First, the problem type may be either site-selection (S) or site-generation (G), depending on whether or not a set of candidate positions is provided.

A problem may either specify m, how many sources have to be opened, or a fixed opening cost F and leave the decision to the algorithm. Therefore, the classic multi-source Weber problem is denoted by Gm and the single-source Weber problem is just a special case of Gm with $m = 1$.

Second, a problem may have a capacity restriction, denoted by c. If this restriction also requires destinations to satisfy all of their demand from a single source, it is a single-source capacity restriction, denoted by $1c$. This makes the single-sourced capacitated MSWP (SSC-MSWP in the literature) $Gm|1c$.

Lastly, a distance limit is indicated by d when present. For instance, the single-source capacitated multi-source Weber problem with fixed opening cost and distance limit, that would be written as SSC-MSWP-FC-DL, becomes just $GF|1c, d$ in our notation. In the literature, distance metrics other than the Euclidean distance have been considered. However, due to the clear prevalence of the Euclidean distance in the literature and in this work, the notation is not extended to specify a distance metric.

1.2 Main variants of the MSWP

Table 1.1 presents the main problems related to the MSWP already addressed in the literature and one source for each problem.

Table 1.1: The works surveyed on the MSWP and its variants.

Variant	Notation	Author	Method
MSWP	Gm	Cooper (1964)	Location-allocation
MSWP-FC	GF	Brimberg, Mladenovic, and Salhi (2004)	Estimate optimal m and solve Gm
C-MSWP-FC	$GF\|c$	Luis, Salhi, and Nagy (2015)	A guided constructive heuristic based on restricted regions and a GRASP
SSC-MSWP	$Gm\|1c$	Öncan (2013)	Alternate location-allocation and very large scale neighborhood search
MSWP-FC-DL	$GF\|d$	Gokbayrak and Kocaman (2017)	Planar set cover followed by location-allocation

This work proposes a new variant, the SSC-MSWP-FC-DL or $GF|1c, d$, in our notation.

1.3 Outline of the thesis

Section 2 presents an overview of the literature on the multi-source Weber problem, its variants, and the main strategies developed to solve it. Section 3 outlines the proposed method for addressing the $GF|1c, d$. Section 4 presents the results obtained with the proposed method. Lastly, Section 5 presents the conclusions and opportunities for future research.

2 LITERATURE REVIEW

Location problems can be categorized according to the location space, which can be continuous, based on a network or discrete, and according to their context, that is, their objectives, constraints or type of facilities involved. Some branches being intensively investigated nowadays are multi-criteria facility location, multi-period facility location, facility location under uncertainty, location-routing, and competitive facility location (LAPORTE; NICKEL; GAMA, 2015, p. 8–9).

The main difference between location and layout problems is that the facilities in location problems are small relative to the space in which they are placed; but in layout problems, the facilities to be located are large relative to the space in which they are positioned (FARAHANI; HEKMATFAR, 2009).

Some authors use "continuous location-allocation" to refer to the multi-source Weber problem. However, because "continuous location-allocation" is similar to the name of a greedy algorithm and is also the name of a broader class of problems, according to Öncan (2013), the more specific name "multi-source Weber problem" has been chosen.

Problems with predetermined candidate locations may be called *site-selecting location problems*, while problems in which the facilities can be located at any point in a continuous space are known as *site-generating location problems*. Following this convention, we are interested in a site-generating problem, which has also been called a greenfield problem by Gokbayrak and Kocaman (2017) because it involves undeveloped sites that have no existing infrastructure and the facilities can be located at any point in a continuous space.

The following subsections present classical algorithms still used to tackle these problems and an overview of the literature on the MSWP with additional constraints.

2.1 Cooper's seminal work

Cooper (1964) describes heuristic methods for Gm. Let $I = [n]$ be set of destinations and $J = [m]$ a set of sources. If the sources have no capacity limit, then no destination has to be supplied by more than one source: the optimal solution consists of assigning each destination to the closest source.

Therefore, the total cost ϕ of supplying n destinations with m sources is given by

$$\phi = \sum_{j \in J} \sum_{i \in I} \alpha_{ij} \psi(i, j)$$

where

$$\psi(i, j) = w_i \left\| \boldsymbol{p}_i - \boldsymbol{x}_j \right\|_2$$

is the distance function, $\boldsymbol{p}_i \in \mathbb{R}^2$ is the position of the i-th destination, $\boldsymbol{x}_j \in \mathbb{R}^2$ is the position of the j-th source, $\alpha_{ij} \in \{0, 1\}$ is a binary decision variable which indicates that destination i is serviced from the source j and $w_i \in \mathbb{R}$ is the weight of destination i. Our intention is to minimize the cost.

Since no client is serviced by more than one facility, $\sum_{j \in J} \alpha_{ij} = 1, \forall i \in I$.

2.1.1 The exact location method

Cooper (1964) presents an optimal solution for the problem of locating a set of sources given a set of destinations and their assignment to these sources. By definition, this algorithm can be used to locate a single source given the positions of the destinations assigned to it. Running this algorithm once for each source ensures all sources have optimal location for a given allocation. The special case with $m = 1$ of the formulation presented above will be used to describe this algorithm. Additionally, because there is only one source and all destinations are assigned to it, α_{ij} is always 1 and will be omitted.

The following equations are necessary first-order conditions for $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2)^T$ to minimize $\phi(\boldsymbol{x})$,

$$\begin{aligned} \frac{\partial \phi}{\partial \boldsymbol{x}_1} &= \sum_{i \in I} \frac{w_i(\boldsymbol{p}_{i1} - \boldsymbol{x}_1)}{\left\| \boldsymbol{p}_i - \boldsymbol{x} \right\|_2} = 0, \\ \frac{\partial \phi}{\partial \boldsymbol{x}_2} &= \sum_{i \in I} \frac{w_i(\boldsymbol{p}_{i2} - \boldsymbol{x}_2)}{\left\| \boldsymbol{p}_i - \boldsymbol{x} \right\|_2} = 0. \end{aligned}$$

Rearranging these equations provides an iterative method. Starting with $\boldsymbol{x}$ as specified in (2.1) and (2.2), and updating it iteratively according to (2.3) and (2.4), converges to the optimal position. In the following equations, ${D_i}^k$ is the value of $\left\| \boldsymbol{p}_i - \boldsymbol{x} \right\|_2$ at the k-th iteration.

$$\boldsymbol{x}_1^0 = \sum_{i \in I} w_i \boldsymbol{p}_{i1}/n \tag{2.1}$$

$$\boldsymbol{x}_2^0 = \sum_{i \in I} w_i \boldsymbol{p}_{i2}/n \tag{2.2}$$

$$\boldsymbol{x}_1^{k+1} = \frac{\sum_{i \in I} w_i \boldsymbol{p}_{i1}/D_i^k}{\sum_{i \in I} w_i/D_i^k} \tag{2.3}$$

$$\boldsymbol{x}_2^{k+1} = \frac{\sum_{i \in I} w_i \boldsymbol{p}_{i2}/D_i^k}{\sum_{i \in I} w_i/D_i^k} \tag{2.4}$$

This iterative method that finds $\boldsymbol{x} \in \mathbb{R}^2$ in order to minimize the weighted transportation cost from a set of n destination positions will be called *the exact location method*. Therefore, as long as all possible assignments, i.e. values of α_{ij}, are tested, the optimal solution for the Gm can be found. For n destinations and m sources this requires running this iterative method $\left\{ {n \atop m} \right\}$ times, where $\left\{ {n \atop m} \right\}$ is the Stirling number of the second kind, that is, the number of ways a set of n labelled objects can be partitioned into m nonempty unlabelled subsets.

Cooper observed that this method always converges in practice. Beck and Sabach (2015) proved its convergence, as long as the anchors (destinations) were avoided. According to Beck and Sabach (2015, p. 9), Cooper's exact location method is a rediscovery of the method proposed in Weiszfeld (1937). However, Weiszfeld's original method only allowed constant weights.

Cooper then proposes four heuristics: the *destination subset algorithm*, the *random destination algorithm*, the *successive approximations algorithm*, and the *alternate location and allocation algorithm*, which are explained in the following subsections.

2.1.2 The destination subset algorithm

This algorithm considers all possible subsets of m destinations as sites at which to locate the sources. After selecting and assigning the sources, the exact location method might be used. According to Cooper (1964), because of the lack of a sharp minimum this method is likely to give very good answers. However, due to the combinatorial nature of this heuristic, it is still very expensive.

2.1.3 The random destination algorithm

This method simply places sources at random destination points. As the author notes, this method can only be as good as the destination subset algorithm. One of the stopping criteria used by the author was finding a solution within a specified number of standard deviations from the mean of the solutions already found, which assumes that the objective function follows a normal distribution.

2.1.4 The successive approximations algorithm

This method uses the destination subset algorithm to place two sources. Then, it adds each remaining source at the destination site where it will improve the objective the most after reassigning the destinations to their cheapest source. The algorithm stops when no source addition decreases the objective function.

2.1.5 The alternate location and allocation algorithm

This algorithm divides the destination set into m subsets of approximately the same size. For each of these subsets the optimal single source location is determined using the exact location method. Then each destination is reassigned to the cheapest source with respect to cost. If the previous step changed the allocation of any destination, the algorithm goes back to using the exact location method to locate the sources. Otherwise, it stops.

Cooper notes that this algorithm is noticeably different from the other three. The main idea behind it, decomposing a location and allocation problem into an allocation-only phase and a location-only phase, is still used today (GOKBAYRAK; KOCAMAN, 2017).

2.2 Weiszfeld's method

Weiszfeld (1937) provided a sequence of steps that was supposed to converge to the optimal solution for the Weber problem. Weiszfeld did not tackle the weighted problem, instead he assumed all weights are equal to 1 (BECK; SABACH, 2015, p. 4). In

2009, the original paper from Weiszfeld was translated with annotations by Frank Plastria in Weiszfeld and Plastria (2009).

The classic Weiszfeld's algorithm relies on a gradient search which requires that the sources never coincide with the anchors (destinations). When the set of anchors is not collinear, the objective function f is strictly convex and thus the optimal solution is unique (BECK; SABACH, 2015). The algorithm works by letting $\boldsymbol{x}_0 \in \mathbb{R}^d \setminus \mathcal{A}$, where $\mathcal{A}$ is the set of anchor points, then by letting $\boldsymbol{x}_{k+1} = T(\boldsymbol{x}_k)$ until $\|\boldsymbol{x}_{k+1} - \boldsymbol{x}_k\| \le \epsilon$, with

$$T(x) = \frac{1}{\sum\limits_{i\in[m]} \frac{w_i}{\|\boldsymbol{x}-\boldsymbol{a}_i\|}} \sum_{i\in[m]} \frac{w_i \boldsymbol{a}_i}{\|\boldsymbol{x} - \boldsymbol{a}_i\|}.$$

The original algorithm is not well defined. Even if the initial vector $\boldsymbol{x}$ does not coincide with an anchor point, the sequence generated may coincide with an anchor point eventually, which will cause divisions by zero. These situations are described in the literature as "getting stuck", and starting points which lead to it are called "bad" starting points (BECK; SABACH, 2015, p. 13).

2.2.1 Modified Weiszfeld's method

In Beck and Sabach (2015, p. 17), a modified Weiszfeld's method that treats reaching anchor points in a "surgical" manner is proposed. This method uses the same formula as Weiszfeld's method when the current point is not an anchor point. However, when the current point coincides with an anchor point, the method will either do nothing (if this anchor point is optimal) or result in another point, with a smaller function value.

The anchor point $\boldsymbol{a}_j$ is optimal if and only if $\|\boldsymbol{R}_j\| \le w_j$, with

$$\boldsymbol{R}_j = \sum_{i=1, i\neq j}^{m} w_i \frac{\boldsymbol{a}_j - \boldsymbol{a}_i}{\|\boldsymbol{a}_i - \boldsymbol{a}_j\|}.$$

With this, the modified Weiszfeld's method can be stated as follows:

$$\boldsymbol{x}_{k+1} = \widetilde{T}(\boldsymbol{x}_k) = \begin{cases} T(\boldsymbol{x}_k), & \boldsymbol{x}_k \notin \mathcal{A}, \\ \boldsymbol{a}_j, & \boldsymbol{x}_k = \boldsymbol{a}_j \text{ and } \|\boldsymbol{R}_j\| \le w_j, \\ S(\boldsymbol{a}_j), & \boldsymbol{x}_k = \boldsymbol{a}_j \text{ and } \|\boldsymbol{R}_j\| > w_j. \end{cases}$$

This leaves us with the requirement of having an operator $S(\boldsymbol{a}_j)$ that produces a new point when the current point is a non-optimal anchor point.

According to Beck and Sabach (2015, p. 18), all of the papers that deal with the modified Weiszfeld's method choose the operator as $S(\boldsymbol{a}_j) = \boldsymbol{a}_j + t_j \boldsymbol{d}_j$, where t_j is a stepsize and

$$\boldsymbol{d}_j = -\frac{\boldsymbol{R}_j}{\|\boldsymbol{R}_j\|}.$$

One possible step size is the one proposed in (RAUTENBACH et al., 2004 apud BECK; SABACH, 2015):

$$t_j = \min\left(\frac{\min\{\|\boldsymbol{a}_j - \boldsymbol{a}_i\| : i \neq j, 1 \leq i \leq m\}}{2}, \frac{\|\boldsymbol{R}_j\| - w_j}{4L(a_j)}\right)$$

with

$$L(\boldsymbol{x}) = \begin{cases} \sum_{i=1}^{m} \frac{w_i}{\|\boldsymbol{x}-\boldsymbol{a}_i\|}, & \boldsymbol{x} \notin \mathcal{A}, \\ \sum_{i=1, i\neq j}^{m} \frac{w_i}{\|\boldsymbol{a}_j-\boldsymbol{a}_i\|}, & \boldsymbol{x} \in \mathcal{A}. \end{cases}$$

This step size proposed by Rautenbach is the one we used in our implementation of the Weiszfeld's method. The reason for choosing it over the alternatives mentioned in Beck and Sabach (2015) is that it is the smallest proposed stepsize. As Weiszfeld's method converges quite quickly, it seems reasonable to use the most conservative stepsize.

As Beck and Sabach (2015, p. 21) point out, the Weber problem can also be solved by more sophisticated methods than Weiszfeld's method, such as Newton methods combined with an active set approach. The problem can also be recast as a second-order cone programming and solved via interior point methods.

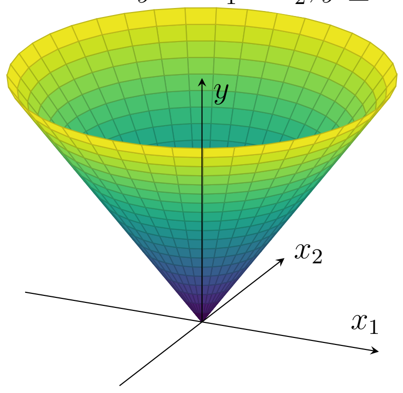
2.2.2 Second-order cone programming

According to Boyd and Vandenberghe (2004, p. 156), a second-order cone program (SOCP) is a problem closely related to quadratic programming. It has the following form:

$$\begin{array}{ll} \text{minimize} & \boldsymbol{f}^T\boldsymbol{x} \\ \text{subject to} & \|\boldsymbol{A}_i\boldsymbol{x} + \boldsymbol{b}_i\|_2 \leq \boldsymbol{c}_i^T\boldsymbol{x} + \boldsymbol{d}_i, \quad i = 1, ..., n, \\ & \boldsymbol{F}\boldsymbol{x} = g, \end{array}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the optimization variable, $\boldsymbol{A}_i \in \mathbb{R}^{n_i \times n}$, and $\boldsymbol{F} \in \mathbb{R}^{p \times n}$.

Figure 2.1: The $y^2 = x_1^2 + x_2^2, y \geq 0$ cone.



The name arises from the second-order cone constraints, which require an affine function to lie in a second-order cone in $\mathbb{R}^{k+1}$. The cone $y^2 = x_1^2 + x_2^2, y \geq 0$ is illustrated in Figure 2.1. In a SOCP, the solution can be required to lie in the convex region bound by a second-order cone, thus $y^2 \geq x_1^2 + x_2^2, y \geq 0$ is a valid constraint in a SOCP.

Second-order conic programs are more general than quadratically-constrained quadratic programs (QCQP) (BOYD; VANDENBERGHE, 2004).

2.2.2.1 Application to the single source Weber problem with distance limits.

The following second-order cone programming formulation, based on Todd (2004), is used to solve the Weber problem with a distance limit d. When a distance limit is not required, Weiszfeld's method can be used.

$$\begin{array}{lll} \text{minimize} & \boldsymbol{w}^T \boldsymbol{\eta} & \\ \text{subject to} & \|\boldsymbol{x} - \boldsymbol{p}_i\|_2 \leq \boldsymbol{\eta}_i, & i = 1, ..., n, \\ & \boldsymbol{\eta}_i \leq d, & i = 1, ..., n. \end{array}$$

In this model, $\boldsymbol{w} \in \mathbb{R}^n_{>0}$ are the weights, auxiliary variables $\boldsymbol{\eta} \in \mathbb{R}^n$ are the distances, $\boldsymbol{x} \in \mathbb{R}^2$ are the decision variables related to the position of the single source, $\boldsymbol{p} \in (\mathbb{R}^2)^n$ are the positions of the destinations, and $d \in \mathbb{R}_{>0}$ is the distance limit.

2.3 The multi-source Weber problem with fixed opening cost (GF)

The MSWP with fixed opening cost is the continuous analogue of the simple plant location problem (SPLP), in which there is a finite set of potential sites on a network, each with an opening cost (BRIMBERG; SALHI, 2005). The literature on this problem is very scarce, according to Luis, Salhi, and Nagy (2015), which could only find two papers about it.

Because the objective function $T(X, W)$ is highly non-convex, the problem falls in the realm of global optimization problems (BRIMBERG; MLADENOVIC; SALHI, 2004; BRIMBERG; SALHI, 2005).

Brimberg, Mladenovic, and Salhi (2004) state that solving multiple instances each with a fixed number of facilities is usually too slow to be used as a solution for the fixed-charge case, as this procedure replaces a difficult problem by $O(n)$ problems of similar difficulty.

In this article there is a proof that the optimal number of facilities opened in the continuous problem might not be equal to the optimal number of facilities opened in the discrete analogue problem. The authors also demonstrate a property which implies that, for very large problems, the optimal number of facilities in the continuous problem is the same as the optimal number of facilities in its discrete analogue.

Their algorithm has three phases. It starts by finding a good approximation of number of sources m, denoted by $\hat{m}$, by using a heuristic to solve the SPLP (in which plants have a fixed opening cost) over the set of destination points. In phase two, Cooper's location-allocation heuristic is used starting from the result of phase one. Phase three then deals with a mechanism that searches for the optimal number of plants m^*. The assumption of this phase is that m^* must be close to $\hat{m}$, and thus can be obtained by a local search starting from $\hat{m}$.

For the discrete initial solutions, the authors use both a perturbation heuristic developed by Salhi (1997) for the simple plant location problem and a variable neighbourhood search heuristic (VNS-4) from Brimberg, Hansen, et al. (2000) to solve the m-median problem over a range of fixed values of m. However, the refining algorithm only slightly improves the discrete solutions. The best-known values are between 0.02$\%$ and 1.53$\%$ better than the best of the two initial discrete solutions. The mean improvement is 0.45$\%$ with a standard deviation of 0.37$\%$.

Brimberg and Salhi (2005) justify that the opening cost of a facility may vary because of the varying cost of land, region-specific government incentives, and different costs to supply products, services and labor to the facility. They assumed that all facilities must be located in one of several non-overlapping zones, where each zone is a closed convex polygon.

The main focus of Brimberg and Salhi (2005) is to solve the single facility case, which leaves the design and comparison of new heuristics to solve the cases with an unspecified number of open facilities to other studies. The inputs from Brimberg, Hansen, et al. (2000) were used for their experiments.

2.4 The capacitated multi-source Weber problem ($Gm|c$)

In this problem, each customer has a demand which has to be fully satisfied and each facility has a maximum capacity. However, a customer may be served by several facilities. Zainuddin and Salhi (2007) propose a perturbation-based heuristic for this problem.

Adding capacities without a single-source allocation constraint turns the allocation problem into the transportation problem, which can be solved optimally in polynomial time. This is the transportation problem, which can be cast as a linear program (HITCHCOCK, 1941).

In Luis, Salhi, and Nagy (2015), three different fixed-charge functions are used: a constant function, which makes the opening cost of all facilities be the same irrespective of their location and size; a zone-based function, similar to what was used in Brimberg and Salhi (2005); and second-order Voronoi polygons, which provide a continuous cost function.

In the latter approach, the plane is tessellated into Voronoi regions using the customers as seed points. The solution algorithm does not need to construct the Voronoi regions, as it only has to calculate the fixed cost for a finite number of candidates locations during the search. Even though zoning is similar to some real-world scenarios, very small changes in the location of a facility can lead to a large change in establishment cost, which might not be realistic.

The algorithm used in Luis, Salhi, and Nagy (2015) is based on a region rejection heuristic from Luis, Salhi, and Nagy (2009). A customer site is randomly selected to become a facility location. Then, an area around this location is declared to be restricted. This prevents facilities from being located too close to each other. This step is repeated until the required number of facilities are located. After having tested different shapes and sizes for the restricted regions, the authors decided to only use circles. After getting m facility locations from this procedure, Cooper's alternating transportation-location method is used. This method takes a set of m open facilities as input and then solves the transportation problem to find the allocation of customers to these facilities. Then, for each facility, its new location is found using the Weiszfeld's method.

Similarly to Brimberg, Mladenovic, and Salhi (2004), Luis, Salhi, and Nagy (2015) do not consider solving for all values of m a valid strategy. Instead, they use a facility addition heuristic. In this addition heuristic, they use only an approximation to the savings in transportation costs when considering adding a new point, instead of using Cooper's method to compute the actual savings. The authors mention that only stopping after two consecutive increments of m worsen the objective function did help avoid local minima. The authors claim a lack of correlation between the quality of the initial solution with the minimum required number of facilities and the final solution.

The GRASP algorithm proposed in their paper retains the addition heuristic and the avoidance concept used in the region-rejection algorithm. The main differences are the use of a GRASP to find the initial solution instead of a constructive heuristic and the use of a weighted pseudorandom selection when adding a new facility to the solution.

For their experiments, the authors used the four datasets from Brimberg, Hansen, et al. (2000), with the addition of fixed costs and capacity. Every facility is set to have the same capacity irrespective of location. The authors found that GRASP gives better or equal solutions for all the instances when compared to region-rejection results.

Lastly, Luis, Salhi, and Nagy (2015) mention other forms of fixed costs worthwhile of research. They suggest using a fixed opening cost function that contains terms inversely proportional to the facility distance to the customers, as it is usually cheaper to locate facilities further away for inhabited areas.

2.5 The single-source capacitated multi-source Weber problem ($Gm|1c$)

In this problem, each customer has a demand which has to be fully satisfied and each facility has a maximum capacity. Differently from $Gm|c$ defined above, in the $Gm|1c$ each customer can only be supplied by a single source.

Gong et al. (1997) proposes an iterative method including location and allocation phases known as Hybrid Evolutionary Method (HEM) to solve the problem. Manzour-al-Ajdad, Torabi, and Eshghi (2012) propose a two phase algorithm that, at phase I, aims to determine proper locations for facilities and, during phase II, pursues an assignment of customers to these facilities. The paper also points out that the number of studies regarding the problem is scarce in spite of the practical relevance of the problem. Lastly, Öncan (2013) considers both the Euclidean and rectilinear distance cases of the $Gm|1c$

and proposes an alternate location and allocation heuristic to solve it.

2.6 The multi-source Weber problem with fixed opening cost and distance limits ($GF|d$)

The MSWP can be extended by limiting the distance between sources and destinations. To the best of our knowledge Gokbayrak and Kocaman (2017) is the only work which tackles this variant.

Gokbayrak and Kocaman (2017) attempt to solve the $Gm|d$ by using a modified Weiszfeld's method which guarantees that the location phase does not violate the distance limit. They also solve a planar set cover problem using the circle intersection points from circles centered at the destinations in order to find more points for their initial candidate set.

2.7 The single-source capacitated multi-source Weber problem with fixed opening cost and distance limits ($GF|1c, d$)

To the best of our knowledge, $GF|1c, d$ has not been analyzed in the literature. It is the main focus of this thesis and can be understood as a refinement of the $GF|d$ as proposed in Gokbayrak and Kocaman (2017) that also takes the capacity limits of real-world sources into consideration.

2.7.1 MISOCP formulation

Problem $GF|1c, d$ can be formulated as the following mixed-integer SOCP (MISOCP) program:

$$
\begin{aligned}
\text{minimize} \quad & \boldsymbol{w}^T \boldsymbol{\tau} + \sum_{j \in J} \boldsymbol{y}_j F \\
\text{subject to} \quad & \|\boldsymbol{x}_j - \boldsymbol{p}_i\|_2 \le \boldsymbol{\eta}_{ij}, && \forall i \in I, \forall j \in J, \\
& \boldsymbol{\eta}_{ij} \le \boldsymbol{\tau}_i + M(1 - \boldsymbol{z}_{ij}), && \forall i \in I, \forall j \in J, \\
& \boldsymbol{\tau}_i \le d, && \forall i \in I, && (2.5) \\
& \boldsymbol{x}_{j1} \le \boldsymbol{x}_{(j+1)1}, && \forall j \in J \setminus \{m\}, && (2.6) \\
& \sum_{i \in I} \boldsymbol{w}_i \boldsymbol{z}_{ij} \le c, && \forall j \in J, && (2.7) \\
& \sum_{j \in J} \boldsymbol{z}_{ij} = 1, && \forall i \in I, && (2.8) \\
& \boldsymbol{y}_j \ge \boldsymbol{z}_{ij}, && \forall i \in I, \forall j \in J, \\
& \boldsymbol{y}_j \ge \boldsymbol{y}_{j+1}, && \forall j \in J \setminus \{m\}. && (2.9)
\end{aligned}
$$

In this model, $\boldsymbol{w} \in \mathbb{R}^n_{>0}$ are the weights, $F \in \mathbb{R}_{>0}$ is the fixed-cost associated with opening a new facility, $\boldsymbol{\eta} \in \mathbb{R}^{n \times m}$ are the distances from the destinations to the sources, and $\boldsymbol{\tau} \in \mathbb{R}^n$ are the distances from the destinations to their assigned sources. Constant $M \in \mathbb{R}$ could be set, for instance, to the diagonal of the bounding box of the set of destinations or any other value sufficiently large to relax the distance limit constraint when there is no assignment. The decision variables related to the positions of the sources are represented by $\boldsymbol{x} \in (\mathbb{R}^2)^n$ and the positions of the destinations are represented by $\boldsymbol{p} \in (\mathbb{R}^2)^n$. In the binary decision matrix $\boldsymbol{z} \in \{0, 1\}^{n \times m}$, 1 indicates that the i-th source is assigned to the j-th destination and 0 indicates the opposite. The distance limit is given by the constant $d \in \mathbb{R}_{>0}$ and the capacity limit is given by the constant $c \in \mathbb{R}_{>0}$.

Constraint (2.5) enforces the distance limit, constraint (2.6) breaks symmetry by requiring the facilities to be ordered by non-increasing x, constraint (2.7) ensures the capacity of the facilities is not exceeded, constraint (2.8) guarantees the demand of each customer is met and constraint (2.9) breaks symmetry by requiring that if p facilities are opened, the first p elements J have to be used.

Even though having different per-facility capacities and distance limits is common in site-selecting problems, this does not make much sense in site-generating problems when all facilities have the same opening cost and can be placed anywhere, which is why in the formulation two constants c and d are used.

This formulation only allows up to n sources to be opened. However, no solution with more than n sources is useful as each of the n destinations can only be assigned to a single source.

Notably, this formulation can be modified by removing constraints. For instance, by setting d to a value larger than the diagonal of the bounding box of the destinations, the distance limit is dropped. Similarly, by setting c to the sum of $\boldsymbol{w}$, the capacity limit is waived.

3 A LOCATION-ALLOCATION METHOD FOR THE SINGLE-SOURCE CAPACITATED MULTI-SOURCE WEBER PROBLEM WITH FIXED OPENING COST AND DISTANCE LIMITS

In this chapter, a new location-allocation algorithm is proposed in order to find feasible solutions for the $GF|1c, d$. The main idea is to obtain a set of candidate positions C of elements from $\mathbb{R}^2$, select a few of these candidate positions to open facilities on and assign the customers to these facilities. A location-allocation phase for refinement in the continuous space is then used to improve the solution.

To simplify the presentation of the algorithms, a problem instance will be defined as a tuple of the form $I = (\boldsymbol{p}, \boldsymbol{w}, c, F, d)$ in which the locations of the destinations are in set $\boldsymbol{p}$, the function $\boldsymbol{w}\colon \boldsymbol{p} \to \mathbb{R}$ maps destinations to their demands, the capacity of a source is c, the fixed opening cost is F and the distance limit is d. A solution consists of two elements: a set of positions for the sources $\boldsymbol{x}$ and an assignment function $a\colon \boldsymbol{p} \to \boldsymbol{x}$. The inverse of a is defined as $a^{-1}\colon \boldsymbol{x} \to \mathcal{P}(\boldsymbol{p})$, where $\mathcal{P}(\boldsymbol{p})$ is the power set of $\boldsymbol{p}$. These can be implemented through an array, which would make evaluating a $O(1)$ and evaluating a^{-1} $O(|\boldsymbol{p}|)$.

A function $\phi(I, \boldsymbol{x}, a)$ that computes the cost of a solution $(\boldsymbol{x}, a)$ for an instance I will also be used. This function can be implemented in $O(|\boldsymbol{p}|)$.

A helper procedure ISFEASIBLE that evaluates whether or not a solution $(\boldsymbol{x}, a)$ for an instance I is feasible will also be used. It can also be implemented in $O(|\boldsymbol{p}| + |\boldsymbol{x}|) = O(|\boldsymbol{p}|)$ by testing the distance limit for each destination while accumulating the demand of each source in an array of length $O(|\boldsymbol{x}|)$.

Another procedure GREEDILYALLOCATE that returns an allocation a when given an instance and a set of destination positions $\boldsymbol{x}$ and can be implemented in $O(|\boldsymbol{p}||\boldsymbol{x}|)$ is also required. It assigns each destination to its closest source.

In our algorithm, shown in Algorithm 1, FINDCANDIDATES generates a set of candidates C for source locations in $\mathbb{R}^2$. SELECTCANDIDATES selects a subset of C denoted by $\boldsymbol{x}$ and assigns each destination to exactly one value from $\boldsymbol{x}$ through the assignment function a. RELOCATE is called to adjust the location of the sources. ALLOCATE is called to adjust the assignment of the destinations. Once the allocation step does not change the allocation of the destinations to the sources, the algorithm stops. These steps are detailed in the following sections.

This algorithm is repeated until a certain number of iterations without improve-

Algorithm 1 The proposed solution method

```
Input a problem instance $I = (\boldsymbol{p}, \boldsymbol{w}, c, F, d)$, a limit of iterations without improvement $\alpha$
Output the locations of the sources $\boldsymbol{x}$ and an assignment function $a$
1:  procedure SOLVE($I$)
2:      $\boldsymbol{x}^* \leftarrow$ undefined
3:      $a^* \leftarrow$ undefined
4:      $i \leftarrow 0$
5:      while $i < \alpha$ do
6:          $C \leftarrow$ FINDCANDIDATES($I, \boldsymbol{x}^*$)
7:          $(\boldsymbol{x}, a) \leftarrow$ SELECTCANDIDATES($I, C$)
8:          $\boldsymbol{x} \leftarrow$ RELOCATE($I, \boldsymbol{x}, a$)
9:          while ALLOCATE($I, \boldsymbol{x}, a$) $\neq a$ do
10:             $a \leftarrow$ ALLOCATE($I, \boldsymbol{x}, a$)
11:             $\boldsymbol{x} \leftarrow$ RELOCATE($I, \boldsymbol{x}, a$)
12:         if $\boldsymbol{x}$ = undefined or $\phi(I, \boldsymbol{x}, a) < \phi(I, \boldsymbol{x}^*, a^*)$ then
13:             $\boldsymbol{x}^* \leftarrow \boldsymbol{x}$
14:             $a^* \leftarrow a$
15:             $i \leftarrow 0$
16:         else
17:             $i \leftarrow i + 1$
18:     return $(\boldsymbol{x}^*, a^*)$
```

ment α is reached. It was empirically determined that 10 was a suitable value of α and, unless otherwise mentioned, all experiments use this value. Other values of α tested when running our implementation on the instances described in Section 4.1 are 100 and 1000. The computational time increases linearly with α and the improvements in the objective function are fairly small.

3.1 Finding candidates

The FINDCANDIDATES procedure computes the union of three sets: the set of destinations $\boldsymbol{p}$, the solution of a planar set cover problem and the position of the sources in the best solution found $\boldsymbol{x}^*$.

3.1.1 Planar set cover problem (PSCP)

Borrowing an idea from Gokbayrak and Kocaman (2017) for the initialization phase, a planar set cover problem is solved. The candidates are the destinations themselves and the intersection points of the circles of radius d centered at the destinations. Since two circles have at most two intersection points, this generates at most $n+2\binom{n}{2} = n^2$ candidates. In practice this number can be much smaller if the distance limit d is small.

3.1.2 Best solution so far

The position of the sources in the best solution so far are added to the candidate set C. In practice, this does not cause a significant increase in solution time and improves the quality of the next solutions.

3.2 Selecting candidates

The initialization step is based on the SUBDROP heuristic proposed in Salhi and Atkinson (1995). SUBDROP utilizes randomly selected subsets of the candidate facility set and drops facilities from the subset in a greedy order until the number of desired facilities is achieved.

3.2.1 Finding the starting solution size

The original SUBDROP heuristic had guidelines for the estimation of the size of the starting solutions based on p (the desired number of facilities). Because in our problem the number of facilities to be opened is a decision variable, those guidelines could not be used. Instead, Algorithm 2 is proposed to find a starting solution size. It performs a stochastic binary search to find the smallest starting size that produces a feasible solution with probability $p \geq 0.75$ after greedy assignment.

The initial solution size should be less than $|C|$ in order to allow for a large number of possible starting solutions and, consequently, a better exploration of the solution space. However, if the initial solution size gets too small, it might be impossible to find a random subset for which greedy assignment produces a feasible solution.

The helper procedure FINDSUCCESSRATE(I, C, t, k) evaluates the number of successful trials in t trials. In this procedure, a trial is a random selection of k candidate sources greedily assigned to their destinations. A trial is successful if, after the greedy assignment, the solution does not violate any of the problem constraints. The number of

Algorithm 2 Find SUBDROP starting solution size

```
    Input the problem instance $I$ and the locations of the candidates $C$
    Output the starting solution size for SUBDROP
 1: procedure FINDSTARTINGSIZE($I, C$)
 2:     $L \leftarrow 1$
 3:     $R \leftarrow |C|$
 4:     while $(L + 1 < R)$ do
 5:         $x \leftarrow \lfloor (L + R)/2 \rfloor$
 6:         if FINDSUCCESSRATE($I, C, 20, x$) $\geq 15$ then
 7:             $R \leftarrow x$
 8:         else
 9:             $L \leftarrow x$
10:     return $L$
```

trials $t = 20$ was empirically determined as a good trade-off between speed and accuracy. Other values tested include 5, 10, 50 and 100.

3.2.2 The modified SUBDROP

The proposed implementation of SELECTCANDIDATES is a modified version of SUBDROP as presented in Algorithm 3. First, it determines the starting solution size by calling FINDSTARTINGSIZE. Then it will perform at least k iterations of the following steps.

1. Select a random subset of the set of candidates of the size determined by the procedure FINDSTARTINGSIZE.
2. Remove sources from this set until the objective function can no longer be improved without violating the constraints.
3. Update the best solution so far if the newly found one is better.

If the initial random subset is not a feasible solution after greedily assigning the destinations to their closest sources, the iteration is discarded and does not count towards k. In two decisions (whether to change to a better source removal and whether to stop removing sources), there is a probability of 75% of making the greedy choice. Introducing these probabilistic locally sub-optimal choices improved the performance of the modified SUBDROP and the variety of the source selections it produces.

A brief complexity analysis of Algorithm 3 shows it is $O(km^3|\boldsymbol{p}|)$. The simplest example of this complexity is line 15, which can be trivially implemented in $O(m|\boldsymbol{p}|)$ and

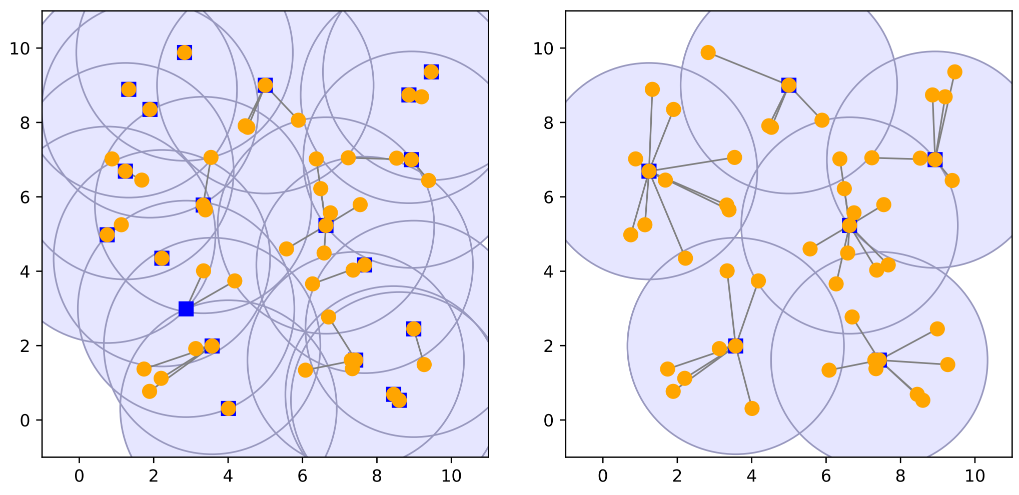
Algorithm 3 The SUBDROP procedure for candidate selection

Input the problem instance I, the locations of the candidates C, and the number of iterations k

Output a set of source locations $\boldsymbol{x}$ and an assignment a

```
1:  procedure SUBDROP($I, C, k$)
2:      $m \leftarrow$ FINDSTARTINGSIZE($I, C$)
3:      $\boldsymbol{x}^* \leftarrow$ undefined
4:      $a^* \leftarrow$ undefined
5:      $i \leftarrow 0$
6:      while $\boldsymbol{x}^*$ = undefined or $i < k$ do
7:          $\boldsymbol{x} \leftarrow$ a random subset of $C$ of size $m$
8:          $a \leftarrow$ GREEDILYALLOCATE($I, \boldsymbol{x}$)
9:          if not ISFEASIBLE($I, \boldsymbol{x}, a$) then
10:             continue
11:         while $|\boldsymbol{x}| > 1$ do
12:             $s \leftarrow$ undefined
13:             for $x \in \boldsymbol{x}$ do
14:                 $\boldsymbol{x}_x \leftarrow \boldsymbol{x} \setminus \{x\}$
15:                 $a_x \leftarrow$ GREEDILYALLOCATE($I, \boldsymbol{x}_x$)
16:                 if ISFEASIBLE($I, \boldsymbol{x}_x, a_x$) then
17:                     if $s$ = undefined then
18:                         $s \leftarrow x$
19:                     else
20:                         $\boldsymbol{x}_s \leftarrow \boldsymbol{x} \setminus \{s\}$
21:                         $a_s \leftarrow$ GREEDILYALLOCATE($I, \boldsymbol{x}_s$)
22:                         if $\phi(I, \boldsymbol{x}_x, a_x) < \phi(I, \boldsymbol{x}_s, a_s)$ then, with probability 0.75
23:                             $s \leftarrow x$
24:             if $s$ = undefined then            ▷ There are no sources that can be removed.
25:                 break
26:             else if $\phi(I, \boldsymbol{x}, a) < \phi(I, \boldsymbol{x}_s, a_s)$ then, with probability 0.75
27:                 break
28:             $\boldsymbol{x} \leftarrow \boldsymbol{x} \setminus \{s\}$
29:             $a \leftarrow$ GREEDILYALLOCATE($I, \boldsymbol{x}$)
30:         if $\boldsymbol{x}^*$ = undefined or $\phi(I, \boldsymbol{x}, a) < \phi(I, \boldsymbol{x}^*, a^*)$ then
31:             $\boldsymbol{x}^* \leftarrow \boldsymbol{x}$
32:             $a^* \leftarrow a$
33:         $i \leftarrow i + 1$
34:     return $\boldsymbol{x}^*, a^*$
```

Figure 3.1: An illustration of the proposed algorithm for selecting candidates. In the left, the random subset of candidates the algorithm started with. In the right, the candidates left at the end of the algorithm.



appears nested in three loops of $O(k)$, $O(m)$ and $O(m)$ iterations, respectively. As k is an input constant which always set to 10 after empirically determining this value as a reasonable trade-off, it will be discarded from the analysis, leaving us with $O(m^3|\boldsymbol{p}|)$. Furthermore, $|\boldsymbol{p}|$ is simply the number of customers n and, as this is a single-source assignment problem, $m \leq n$, which simplifies the expression to $O(n^4)$, which was prohibitively slow in practice for n as large as 1060.

Therefore, a data structure which can exploit the properties of this problem is required to make Algorithm 3 useful. The data structure we used consists of building a list of sources for each destination and sorting these lists by transportation cost in non-decreasing order. This can be done in $O(n^2 \log n)$, which is the complexity of ordering n lists of $O(n)$ elements with an $O(n \log n)$ sorting algorithm.

With this structure, it is possible to implement Algorithm 3 in $O(n^2 \log n)$. This happens because determining the feasibility and the objective function of the solution after removing a single source can be calculated for all sources in $O(n)$ time (by looking at the first two elements of each list). Updating this structure requires removing the first elements of at most n lists, which can also be done in amortized $O(n)$ time. Because the loop in which this query and update are inserted is bound to $O(n)$ iterations, the actual procedure of dropping sources takes $O(n(n+n)) = O(n^2)$ time and the overall complexity is given by the complexity of building our acceleration structure.

Figure 3.1 gives an example of an execution of Algorithm 3.

3.2.3 Alternatives to the proposed SELECTCANDIDATES method

An alternative to the proposed heuristic is to solve the single-source capacitated facility location problem with distance limits (SSC-FLP-DL or $SF|1c, d$). This is a MILP which is fairly expensive to solve.

The $SF|d$ is used in Gokbayrak and Kocaman (2017) for candidate selection. Solving it using CPLEX is fairly fast. Because the same is not true for the $SF|1c, d$, there is a need for a heuristic alternative (such as Algorithm 3) that can filter the candidate set C in practical time when single-source capacity constraints are introduced.

3.3 Location algorithm

RELOCATE is called to solve the distance-limited Weber problem ($Gm|d$ with $m = 1$) that arises from trying to minimize the transportation cost from a single source to the destinations that are currently assigned to it. It only changes the position of the sources. This is done by solving the program described in Section 2.2.2.1 once for each source. These programs can usually be solved in less than a second.

3.4 Allocation algorithm

ALLOCATE is called to solve the $SF|1c, d$ that arises from trying to minimize the overall transportation cost by redistributing the destinations between the sources. It only changes the assignment of the destinations.

It is solved using the greedy algorithm presented as Algorithm 4 that works by enumerating a set C of all changes that can improve the objective function and applying the changes that do not violate the problem constraints in a greedy order.

An alternative to this greedy strategy is to cast this allocation problem as a generalized assignment problem (GAP) and use a commercial solver to solve it to optimality. This will always result in an allocation at least as good as the one produced by Algorithm 4. However, the solution after the location-allocation loop stops is not guaranteed to be as good as the solution when using Algorithm 4 to implement ALLOCATE.

Algorithm 4 Greedy reallocation algorithm

Input the problem instance $I = (\boldsymbol{p}, \boldsymbol{w}, c, F, d)$, the locations of the sources $\boldsymbol{x}$ and an assignment relation a

Output the new assignment a

```
 1: procedure ALLOCATE($I, \boldsymbol{x}, a$)
 2:     $C \leftarrow \emptyset$
 3:     for $p \in \boldsymbol{p}, x \in \boldsymbol{x}$ do
 4:         $\Delta \leftarrow \|a(p) - p\|_2 - \|x - p\|_2$
 5:         if $\Delta > 0$ then
 6:             $C \leftarrow C \cup \{(\Delta, p, x)\}$
 7:     moved $\leftarrow \{\}$
 8:     for $(\Delta, p, x) \in C$ ordered by non-increasing $\Delta$ do
 9:         if $p \notin$ moved then
10:             if $\boldsymbol{w}(p) + \sum_{k \in a^{-1}(x)} \boldsymbol{w}(k) \le c$ then
11:                 $a(p) \leftarrow x$
12:                 moved $\leftarrow$ moved $\cup \{p\}$
13:     return $a$
```

$$
\begin{aligned}
GAP = \text{minimize} \quad & \boldsymbol{w}^T \boldsymbol{\tau} \\
\text{subject to} \quad & \|\boldsymbol{x}_j - \boldsymbol{p}_i\|_2 \, \boldsymbol{z}_{ij} \le \boldsymbol{\tau}_i, && \forall i \in I, \forall j \in J, && (3.1) \\
& \boldsymbol{\tau}_i \le d, && \forall i \in I, && (3.2) \\
& \sum_{i \in I} \boldsymbol{w}_i \boldsymbol{z}_{ij} \le c, && \forall j \in J, && (3.3) \\
& \sum_{j \in J} \boldsymbol{z}_{ij} = 1, && \forall i \in I. && (3.4)
\end{aligned}
$$

In this model, $\boldsymbol{w} \in \mathbb{R}^n_{>0}$ are the weights and $\boldsymbol{\tau} \in \mathbb{R}^n$ are the distances from the destinations to their assigned sources. The positions of the sources are problem inputs represented by $\boldsymbol{x} \in (\mathbb{R}^2)^n$ and the positions of the destinations are represented by $\boldsymbol{p} \in (\mathbb{R}^2)^n$. In the binary decision matrix $\boldsymbol{z} \in \{0, 1\}^{n \times m}$, 1 indicates that the i-th source is assigned to the j-th destination and 0 indicates the opposite. The distance limit is given by the constant $d \in \mathbb{R}_{>0}$ and the capacity limit is given by the constant $c \in \mathbb{R}_{>0}$.

Constraint (3.1) makes $\boldsymbol{\tau}_i$ equal to the distance between customer i and its source, constraint (3.2) ensures that the distance limit is not violated, constraint (3.3) ensures the capacity of the facilities is not exceeded, constraint (3.4) guarantees the demand of each customer is met.

The classic GAP formulation does not have distance constraints and it can be used to solve this allocation problem by adding constraints of the form $\boldsymbol{z}_{ij} = 0$ when destination i is too far away from source j. However, the formulation presented here is substantially faster on CPLEX than the classical formulation and several values from $\boldsymbol{z}$ set to zero.

3.5 Argument for feasibility

Algorithm 1 is guaranteed to find a solution for the $GF|1c, d$ if one exists as long as $\boldsymbol{p} \subseteq C$. All variations of the method proposed in this work include all elements of $\boldsymbol{p}$ in C. Therefore, as long as SELECTCANDIDATES outputs a feasible solution, which is always the case for our modified SUBDROP when $\boldsymbol{p} \subseteq C$, feasibility is maintained until the end. RELOCATE does not break feasibility as it does not violate the distance limit when relocating and does not change the assignment function a. The proposed ALLOCATE procedure does not break feasibility if the input state was feasible, as it will never commit moves that leave the current solution in an infeasible state.

4 EXPERIMENTAL RESULTS

In this chapter, we introduce the instances used to test the proposed method, the experimental methodology and our results.

4.1 Instances

A complete input for the $GF|1c, d$ requires a set of points for the destinations, a set of associated weights, a capacity limit c, a fixed opening cost F and a distance limit d. As this is a new problem, the instances from the literature provide only some of these values.

The base instances used from the literature include the 50-customer instance from (EILON; WATSON-GANDY; CHRISTOFIDES, 1971 apud ÖNCAN, 2013), the 287-customer instance from (BONGARTZ; CALAMAI; CONN, 1994 apud GOKBAYRAK; KOCAMAN, 2017), the 654-customer and the 1060-customer instances from (REINELT, 1991 apud GOKBAYRAK; KOCAMAN, 2017). Among these, only the 287-customer instance has non-unitary weights. Its weights are integral and range from 1 to 698.

In order to be able to compare the obtained results with published results, the choice of c, F and d is based on what has already been used in the literature. In this work, we extend the 75 instances used by Gokbayrak and Kocaman (2017) to have capacity limits.

Gokbayrak and Kocaman (2017) do not use the 50-customer base instance. The distance limit for this customer set was defined based on the concept of a customer area, calculated as the area of the bounding box of the customers divided by the number of customers. Then, by interpreting this area as the area of a circle, a customer radius r_c can be calculated. In the 50-customer set, this radius is roughly 0.728459. This value is quite restrictive, as it corresponds to the radius for a single customer, and interesting instances should allow including multiple customers per source. Therefore, when creating instances, the radii $r_c \begin{bmatrix} 2 & 3 & 4 & 5 & 6 \end{bmatrix}$ were used. In order to simplify presentation and input, these values were rounded to 2 decimal places. A visualization of these radii around the destination points is provided in Figure 4.1.

Given that Gokbayrak and Kocaman (2017) do not provide facility capacities, the capacities used in other works were considered. Manzour-al-Ajdad, Torabi, and Eshghi (2012) do not include fixed opening costs in the formulation and, because of this, set the capacity to the average weight $\sum \boldsymbol{w}_i/m$ when m facilities are going to be opened. Öncan (2013) uses the same capacities as Manzour-al-Ajdad, Torabi, and Eshghi (2012).

Because in our formulation the number of opened sources is a decision variable, it is not possible to use the same capacities as these authors. It is important to note that letting either the capacity limit or the distance limit be much stricter than the other is undesired as this would make one of the proposed constraints irrelevant for testing the method. Therefore, given that the literature has distance limits and a method for obtaining distance limits when they are not available from the literature has been proposed, the capacity limits will be derived after the derivation of the distance limits. This facilitates finding capacity limits which are not too strict or too relaxed when compared to the distance limits.

By considering opening a source at a destination site i and assigning all destinations reachable from this site to this hypothetical source, a maximum demand D_i at this site can be calculated. A solution with substantial coverage overlap is possibly the result of the capacity constraint being much stricter than the distance limit constraint. Given this observation, a fair capacity limit should not require substantial coverage overlap. In the ideal case, when there is no overlap, the capacity for a site opened at the i-th destination point has to be D_i. However, some destination points might be subject to more demand than others, which leads to the following observation: by letting the capacity limit be equal to the average of the maximum demands at the candidate sites, some sites will have sources with unused capacity and others will have sources with capacity overflow. Therefore, setting the capacity constraint to $c = \sum_{i=1}^{n} D_i/n$ should make it both a non-trivial constraint and a constraint not much tighter than the distance limit.

Lastly, for the 50-customer set, the fixed opening costs also had to be determined. It was stipulated that the first case would have a fixed opening cost F_1 from the set $\{0.5, 1.0, 1.5, 2.0, 2.5\}$ and all other cases would be have a fixed opening cost of $F_k = (c_k/c_1)(d_k/d_1)F_1$, which makes the cost of opening a facility directly proportional to its capacity and coverage radius.

This creates the 100 instances shown in Table 4.1.

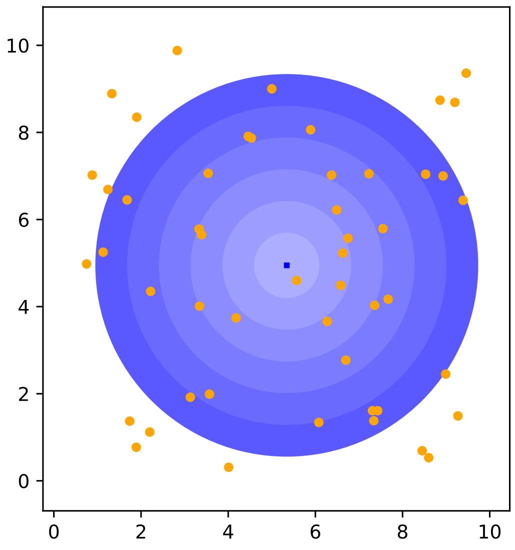
4.2 Experimental methodology

The proposed method has been implemented in C++ and compiled using GCC 9.2.1 with the `-O3`, `-DNDEBUG` and `-std=gnu++2a` flags. CPLEX 12.9 was used in

Table 4.1: The instances used to test the proposed method.

n	c	F	d
50	4	0.50	1.46
50	8	1.43	2.19
50	11	2.85	2.91
50	17	5.25	3.64
50	22	8.37	4.37
50	4	1.00	1.46
50	8	2.86	2.19
50	11	5.71	2.91
50	17	10.49	3.64
50	22	16.74	4.37
50	4	1.50	1.46
50	8	4.29	2.19
50	11	8.56	2.91
50	17	15.74	3.64
50	22	25.11	4.37
50	4	2.00	1.46
50	8	5.72	2.19
50	11	11.42	2.91
50	17	20.98	3.64
50	22	33.49	4.37
50	4	2.50	1.46
50	8	7.15	2.19
50	11	14.27	2.91
50	17	26.23	3.64
50	22	41.86	4.37
287	1410	50.00	5.00
287	3886	50.00	10.00
287	5472	50.00	15.00
287	6090	50.00	20.00
287	6220	50.00	25.00
287	1410	100.00	5.00
287	3886	100.00	10.00
287	5472	100.00	15.00
287	6090	100.00	20.00
287	6220	100.00	25.00
287	1410	200.00	5.00
287	3886	200.00	10.00
287	5472	200.00	15.00
287	6090	200.00	20.00
287	6220	200.00	25.00
287	1410	500.00	5.00
287	3886	500.00	10.00
287	5472	500.00	15.00
287	6090	500.00	20.00
287	6220	500.00	25.00
287	1410	5000.00	5.00
287	3886	5000.00	10.00
287	5472	5000.00	15.00
287	6090	5000.00	20.00
287	6220	5000.00	25.00
654	55	1000.00	200.00
654	79	1000.00	400.00
654	96	1000.00	600.00
654	116	1000.00	800.00
654	129	1000.00	1000.00
654	55	2000.00	200.00
654	79	2000.00	400.00
654	96	2000.00	600.00
654	116	2000.00	800.00
654	129	2000.00	1000.00
654	55	5000.00	200.00
654	79	5000.00	400.00
654	96	5000.00	600.00
654	116	5000.00	800.00
654	129	5000.00	1000.00
654	55	10000.00	200.00
654	79	10000.00	400.00
654	96	10000.00	600.00
654	116	10000.00	800.00
654	129	10000.00	1000.00
654	55	15000.00	200.00
654	79	15000.00	400.00
654	96	15000.00	600.00
654	116	15000.00	800.00
654	129	15000.00	1000.00
1060	4	1000.00	200.00
1060	9	1000.00	400.00
1060	15	1000.00	600.00
1060	23	1000.00	800.00
1060	33	1000.00	1000.00
1060	4	2000.00	200.00
1060	9	2000.00	400.00
1060	15	2000.00	600.00
1060	23	2000.00	800.00
1060	33	2000.00	1000.00
1060	4	5000.00	200.00
1060	9	5000.00	400.00
1060	15	5000.00	600.00
1060	23	5000.00	800.00
1060	33	5000.00	1000.00
1060	4	10000.00	200.00
1060	9	10000.00	400.00
1060	15	10000.00	600.00
1060	23	10000.00	800.00
1060	33	10000.00	1000.00
1060	4	15000.00	200.00
1060	9	15000.00	400.00
1060	15	15000.00	600.00
1060	23	15000.00	800.00
1060	33	15000.00	1000.00

Figure 4.1: The customer radius (the lighter disk) and the five radii used to create instances from the 50-customer base instance.



deterministic mode via the Concert API for solving all mathematical programs.

All real values were represented using 64-bit IEC 559/IEEE 754 floating point numbers. Floating point comparison was always performed using a relative difference r. Therefore, two floating point values a and b were considered equal if, and only if, `boost::math::relative_difference(a, b) < r`. In mathematical terms, when there are no zeros, infinites, NaNs, and the values are of the same sign, two values a and b are equal when $|(a-b)/\min\{a,b\}| < r$.

For the solving algorithms, $r = 10^{-5}$ was used. For validating the solutions generated by external solvers and the C++ implementation, $r = 10^{-4}$ was used. The higher tolerance for validation is required because the solutions CPLEX provides for SOCPs often violate the stricter tolerance of 10^{-5}.

The SplitMix pseudorandom number generator (PRNG) proposed in Steele Jr., Lea, and Flood (2014) is the only pseudorandom number generator used. It was always initialized with 1. This decision was made because our algorithm has no need for unpredictability and only needs a fast, cache-friendly PRNG with a fairly large period.

In the following tables, only the best value obtained and the relative differences to it are presented in each row to facilitate comparison. Time is always presented in seconds and is denoted by t. The relative difference was computed as $(\phi - \phi_b)/\phi_b$ where ϕ is the value of the objective function and ϕ_b is the best value for that instance in the table. The last row of the table presents the average relative difference and the average time for each set of results.

4.3 Platform

The platform used for the experiments was an AMD FX-8320E @ 3.2 GHz with 32 GiB of DDR3 RAM. The operating system was OpenSUSE Tumbleweed (Linux 5.3.9). All experiments were conducted using a single thread.

4.4 Solver results

When CPLEX was used to solve a MISOCP, it was limited to 10^4 seconds and 24 GiB of memory for the search tree.

All instances with $n = 50$ were tested on CPLEX using the mathematical program described in Section 2.7.1. CPLEX never found an optimal solution: it timed-out in 24 of the 25 instances and ran out of memory in one instance after 7620 seconds. These results are presented in Table 4.2.

The first and last instances of the groups with $n \in \{287, 654, 1060\}$ were tested and CPLEX did not find any solution within the 10^4 seconds time limit. The other instances with these values of n were not tested, as it is unlikely that the instances between the most distance-limited and the least distance-limited instances would perform any differently.

4.5 Results on the $GF|d$

After removing the single-source capacity constraint, our problem becomes the one analyzed in Gokbayrak and Kocaman (2017). As mentioned in Section 3.2.3, if there are no capacities, the candidate selection step can be solved by a commercial solver in practical time. This is preferred as it is guaranteed to be at least as good as the modified SUBDROP, which does not have any optimality guarantees.

Because CPLEX was running in deterministic mode, the optimal solution to the facility location problem in SELECTCANDIDATES was always the same. Therefore, it made sense to set α (the limit of iterations without improvement) to 1 in these experiments.

Gokbayrak and Kocaman (2017) use a dual 2.4 GHz Intel Xeon E5-2630 v3 CPU server with 64GB RAM and CPLEX 12.7 in parallel mode using up to 32 threads.

Gokbayrak and Kocaman (2017) also tried using 4000 grid nodes and 4000 ran-

Table 4.2: The results for the instances with $n = 50$ destinations using CPLEX and the proposed algorithm.

					MISOCP		Proposed algorithm	
n	c	F	d	Best	R. Diff. (%)	t	R. Diff. (%)	t
50	4	0.50	1.46	22.93	165.2	10000.3	0.0	2.5
50	8	1.50	2.19	49.35	97.4	10000.3	0.0	4.0
50	11	2.74	2.91	68.38	52.2	10000.3	0.0	3.5
50	17	5.30	3.64	91.73	102.7	10000.3	0.0	1.5
50	22	8.23	4.37	110.48	58.9	10000.4	0.0	2.1
50	4	1.00	1.46	38.30	103.8	10000.3	0.0	2.0
50	8	3.00	2.19	71.06	41.9	10000.3	0.0	3.2
50	11	5.48	2.91	92.86	96.0	10000.3	0.0	1.2
50	17	10.60	3.64	124.57	49.9	10000.3	0.0	2.5
50	22	16.46	4.37	149.99	58.8	10000.3	0.0	2.0
50	4	1.50	1.46	49.14	91.8	10000.3	0.0	3.3
50	8	4.50	2.19	86.70	46.5	10000.3	0.0	2.3
50	11	8.22	2.91	110.29	47.6	10001.2	0.0	1.3
50	17	15.89	3.64	147.76	22.5	10000.3	0.0	1.9
50	22	24.69	4.37	180.49	29.2	10000.3	0.0	0.8
50	4	2.00	1.46	58.27	83.3	7620.4	0.0	3.2
50	8	6.00	2.19	100.31	47.6	10001.4	0.0	2.4
50	11	10.96	2.91	126.73	28.3	10000.9	0.0	1.5
50	17	21.19	3.64	168.96	21.3	10000.3	0.0	1.4
50	22	32.92	4.37	205.18	27.4	10000.2	0.0	0.8
50	4	2.50	1.46	66.57	56.8	10000.4	0.0	3.8
50	8	7.50	2.19	112.05	62.3	10000.4	0.0	3.4
50	11	13.70	2.91	143.17	29.2	10000.3	0.0	2.2
50	17	26.49	3.64	190.16	44.1	10000.3	0.0	1.3
50	22	41.16	4.37	229.90	9.0	10000.3	0.0	0.8
Averages					58.9	9905.2	0.0	2.2

Table 4.3: Comparison between the proposed method and the results from Gokbayrak and Kocaman (2017) for the $n = 287$ set.

				Gokbayrak (2017)		Proposed algorithm	
n	F	d	Best	R. Diff. (%)	t	R. Diff. (%)	t
287	50	5	4021.66	0.3	1.0	0.0	4.9
287	50	10	3926.00	0.0	10.0	0.1	32.7
287	50	15	3885.64	0.1	16.0	0.0	31.9
287	50	20	3884.62	0.1	17.0	0.0	33.1
287	50	25	3881.05	0.1	18.0	0.0	45.4
287	100	5	6255.21	0.3	1.0	0.0	4.1
287	100	10	5857.00	0.0	8.0	0.1	30.8
287	100	15	5764.26	0.1	15.0	0.0	32.2
287	100	20	5764.26	0.1	20.0	0.0	33.5
287	100	25	5710.44	0.1	29.0	0.0	44.6
287	200	5	9608.36	0.2	1.0	0.0	5.9
287	200	10	8564.37	0.6	12.0	0.0	30.8
287	200	15	8347.59	0.1	21.0	0.0	36.3
287	200	20	8297.59	0.1	24.0	0.0	38.7
287	200	25	8149.60	0.1	23.0	0.0	45.1
287	500	5	15979.97	0.2	1.0	0.0	5.0
287	500	10	13105.14	0.9	24.0	0.0	33.0
287	500	15	12287.61	0.0	36.0	0.0	32.1
287	500	20	11999.59	0.2	30.0	0.0	33.2
287	500	25	11616.66	0.2	26.0	0.0	36.7
287	5000	5	82867.00	0.0	1.0	0.0	4.2
287	5000	10	47221.30	2.6	32.0	0.0	31.1
287	5000	15	35554.56	0.1	37.0	0.0	33.0
287	5000	20	27483.88	0.3	58.0	0.0	35.0
287	5000	25	26637.62	0.7	43.0	0.0	37.9
Averages				0.3	20.2	0.0	29.2

dom circle intersection points (CIPS) to augment the candidate set instead of the planar set cover on the instances with $n = 1060$. Using 4000 grid nodes was, on average, 6.13% worse and 269 times slower. Using 4000 random CIPS was, on average, 6.76% worse and 2123 times slower. Therefore, our tables contain their results using the planar set cover for augmentation, as it is their best method overall and is the only one reasonably close to ours in execution time.

The results of our method on the $GF|d$ are shown in Tables 4.3, 4.4 and 4.5. From these results, the proposed algorithm for the $GF|d$ seems to be competitive with the one proposed in Gokbayrak and Kocaman (2017) albeit slightly slower. This performance disadvantage may be caused by the difference in computing power between the platforms.

Table 4.4: Comparison between the proposed method and the results from Gokbayrak and Kocaman (2017) for the $n = 654$ set.

				Gokbayrak (2017)		Proposed algorithm	
n	F (k)	d	Best	R. Diff. (%)	t	R. Diff. (%)	t
654	1	200	78190.84	0.6	18.0	0.0	15.5
654	1	400	75166.89	0.2	9.0	0.0	25.6
654	1	600	74686.00	0.0	16.0	0.4	31.5
654	1	800	73853.48	0.2	21.0	0.0	40.6
654	1	1000	73853.48	0.2	18.0	0.0	57.5
654	2	200	120009.01	0.3	24.0	0.0	14.8
654	2	400	108167.00	0.0	11.0	0.1	24.7
654	2	600	103653.08	0.1	25.0	0.0	29.9
654	2	800	102525.75	0.1	17.0	0.0	39.2
654	2	1000	102136.67	0.1	18.0	0.0	57.0
654	5	200	237330.98	0.2	22.0	0.0	15.1
654	5	400	184653.00	0.0	26.0	0.7	23.6
654	5	600	168869.12	0.3	35.0	0.0	29.9
654	5	800	160411.70	0.6	43.0	0.0	38.0
654	5	1000	155150.78	0.2	20.0	0.0	55.5
654	10	200	417330.98	0.1	20.0	0.0	15.3
654	10	400	284128.00	0.0	27.0	0.5	24.7
654	10	600	248739.10	0.2	52.0	0.0	33.3
654	10	800	227474.96	1.0	50.0	0.0	50.9
654	10	1000	219388.19	0.2	28.0	0.0	51.9
654	15	200	597330.98	0.1	22.0	0.0	15.3
654	15	400	378753.00	0.0	62.0	0.3	24.4
654	15	600	323739.10	0.3	58.0	0.0	33.7
654	15	800	292987.57	0.9	82.0	0.0	41.5
654	15	1000	279219.47	0.2	24.0	0.0	51.4
Averages				0.2	29.9	0.1	33.6

Table 4.5: Comparison between the proposed method and the results from Gokbayrak and Kocaman (2017) for the $n = 1060$ set.

				Gokbayrak (2017)		Proposed algorithm	
n	F (k)	d	Best	R. Diff. (%)	t	R. Diff. (%)	t
1060	1	200	431737.85	0.7	14.0	0.0	23.5
1060	1	400	370266.82	0.0	5.0	0.0	28.9
1060	1	600	363270.00	0.0	5.0	0.1	48.0
1060	1	800	362127.00	0.0	5.0	0.1	147.4
1060	1	1000	362120.00	0.0	14.0	0.1	80.9
1060	2	200	730737.85	0.7	4.0	0.0	23.5
1060	2	400	518610.88	0.8	6.0	0.0	29.0
1060	2	600	487723.00	0.0	6.0	0.2	48.4
1060	2	800	483083.00	0.0	6.0	0.1	147.1
1060	2	1000	483009.00	0.0	12.0	0.1	73.6
1060	5	200	1627737.85	0.7	4.0	0.0	23.5
1060	5	400	906131.50	0.0	3.0	0.0	28.5
1060	5	600	746502.00	0.0	7.0	0.5	48.5
1060	5	800	710816.61	0.2	12.0	0.0	148.7
1060	5	1000	705568.00	0.0	19.0	0.0	80.8
1060	10	200	3122550.56	0.7	4.0	0.0	34.4
1060	10	400	1541563.00	0.0	3.0	0.3	28.5
1060	10	600	1132433.78	0.2	7.0	0.0	47.6
1060	10	800	1001892.34	0.4	376.0	0.0	206.8
1060	10	1000	960718.00	0.0	208.0	0.3	104.5
1060	15	200	4617550.56	0.7	4.0	0.0	34.5
1060	15	400	2176563.00	0.0	4.0	0.4	28.5
1060	15	600	1497422.68	0.0	6.0	0.0	47.5
1060	15	800	1272103.36	0.2	47.0	0.0	294.5
1060	15	1000	1170355.61	0.0	370.0	0.0	955.6
Averages				0.2	46.0	0.1	110.5

4.6 The effect of optimal allocation

In this section, the effect of using an optimal allocation step during the location-allocation loop on the solution quality is analyzed.

Based on the results shown in Tables 4.6 to 4.9, performing optimal allocation in the location-allocation loop does not seem to improve solution quality noticeably when compared to Algorithm 4. Additionally, finding the optimal solution through CPLEX does not seem to be a practical alternative as it takes too long in some difficult instances. This required CPLEX to be limited to 100 seconds for each GAP instance it had to solve in these experiments.

The instances with $n = 287$ are the only ones with non-uniform $\boldsymbol{w}$. These are also the instances where using an optimal allocation step improves the results the most. This is a possible indication that the proposed greedy allocation method described in Algorithm 4 could be improved to better handle varying demands.

Table 4.6: The results of the proposed method when using optimal allocation on the instances with $n = 50$ customers.

					Greedy allocation		Optimal allocation	
n	c	F	d	Best	R. Diff. (%)	t	R. Diff. (%)	t
50	4	0.50	1.46	23.45	0.0	0.4	0.0	0.5
50	8	1.50	2.19	49.35	0.0	0.6	0.0	0.9
50	11	2.74	2.91	68.38	0.0	0.6	0.0	0.9
50	17	5.30	3.64	91.73	0.0	0.3	0.0	0.5
50	22	8.23	4.37	110.48	0.0	0.4	0.0	1.0
50	4	1.00	1.46	38.30	0.0	0.6	0.0	0.8
50	8	3.00	2.19	71.06	0.0	0.5	0.0	0.7
50	11	5.48	2.91	92.86	0.0	0.2	0.0	0.3
50	17	10.60	3.64	124.57	0.0	0.5	0.0	0.8
50	22	16.46	4.37	149.99	0.0	0.4	0.0	1.1
50	4	1.50	1.46	49.49	0.0	0.5	0.0	0.7
50	8	4.50	2.19	86.70	0.0	0.4	0.0	0.5
50	11	8.22	2.91	110.29	0.0	0.3	0.0	0.3
50	17	15.89	3.64	147.75	0.0	0.4	0.0	0.6
50	22	24.69	4.37	180.49	0.0	0.2	0.0	0.5
50	4	2.00	1.46	58.62	0.0	0.4	0.0	0.5
50	8	6.00	2.19	100.31	0.0	0.4	0.0	0.5
50	11	10.96	2.91	126.73	0.0	0.3	0.0	0.3
50	17	21.19	3.64	168.96	0.0	0.3	0.0	0.5
50	22	32.92	4.37	205.18	0.0	0.2	0.0	0.4
50	4	2.50	1.46	66.81	0.0	0.6	0.0	0.7
50	8	7.50	2.19	112.05	0.0	0.5	0.0	0.6
50	11	13.70	2.91	143.17	0.0	0.4	0.0	0.5
50	17	26.49	3.64	190.16	0.0	0.3	0.0	0.5
50	22	41.16	4.37	229.90	0.0	0.2	0.0	0.4
Averages					0.0	0.4	0.0	0.6

Table 4.7: The results of the proposed method when using optimal allocation on the instances with $n = 287$ customers.

					Greedy allocation		Optimal allocation	
n	c	F	d	Best	R. Diff. (%)	t	R. Diff. (%)	t
287	1410	50	5	4236.13	0.5	8.3	0.0	31.0
287	3886	50	10	4352.40	7.5	32.6	0.0	98.4
287	5472	50	15	4515.47	0.3	28.5	0.0	42.7
287	6090	50	20	4771.46	6.4	25.1	0.0	81.7
287	6220	50	25	6012.07	5.2	25.9	0.0	955.9
287	1410	100	5	6559.38	1.1	7.7	0.0	22.2
287	3886	100	10	6236.68	0.0	33.2	1.4	42.6
287	5472	100	15	6174.16	1.8	28.3	0.0	51.4
287	6090	100	20	6400.29	0.8	27.1	0.0	95.0
287	6220	100	25	6945.40	6.7	28.3	0.0	2353.3
287	1410	200	5	10150.38	1.2	8.0	0.0	33.7
287	3886	200	10	8752.70	1.8	35.1	0.0	45.9
287	5472	200	15	8488.58	0.8	27.1	0.0	39.5
287	6090	200	20	8392.56	2.0	27.5	0.0	145.8
287	6220	200	25	8922.57	0.0	26.8	6.7	848.4
287	1410	500	5	18015.27	0.5	10.3	0.0	36.7
287	3886	500	10	13223.87	0.0	32.4	0.3	38.6
287	5472	500	15	12249.27	0.1	30.0	0.0	44.8
287	6090	500	20	12015.12	0.0	26.3	2.0	123.3
287	6220	500	25	11680.26	0.0	24.0	1.5	1265.6
287	1410	5000	5	121289.24	0.0	17.8	3.9	21.6
287	3886	5000	10	51233.32	5.5	32.1	0.0	39.0
287	5472	5000	15	37643.85	0.0	29.5	0.0	52.7
287	6090	5000	20	29296.90	0.0	23.9	0.0	49.4
287	6220	5000	25	26637.62	0.0	23.3	0.0	32.2
Averages					1.7	24.8	0.6	263.7

Table 4.8: The results of the proposed method when using optimal allocation on the instances with $n = 654$ customers.

					Greedy allocation		Optimal allocation	
n	c	F (k)	d	Best	R. Diff. (%)	t	R. Diff. (%)	t
654	55	1	200	79917.61	0.0	48.3	0.0	50.4
654	79	1	400	75561.46	0.0	64.6	0.6	63.0
654	96	1	600	74373.49	0.3	38.9	0.0	84.0
654	116	1	800	73858.75	0.0	45.3	0.0	64.5
654	129	1	1000	73859.44	0.0	61.2	0.1	67.1
654	55	2	200	126477.79	0.0	39.3	0.0	54.4
654	79	2	400	110090.49	0.4	46.5	0.0	115.8
654	96	2	600	103703.64	0.0	30.9	0.0	38.3
654	116	2	800	102524.62	0.0	33.1	0.0	42.3
654	129	2	1000	102367.65	0.0	44.2	0.0	53.7
654	55	5	200	253714.63	0.0	62.3	1.8	86.6
654	79	5	400	196207.75	0.0	51.4	0.0	63.3
654	96	5	600	171558.07	0.0	62.4	0.0	78.0
654	116	5	800	162373.67	0.2	47.1	0.0	56.3
654	129	5	1000	155150.66	0.0	53.6	0.0	55.4
654	55	10	200	468714.63	0.0	62.2	2.1	86.5
654	79	10	400	336160.35	0.0	59.2	0.0	74.5
654	96	10	600	266544.08	0.0	46.3	0.0	55.8
654	116	10	800	237396.10	0.0	31.7	0.0	34.6
654	129	10	1000	219388.19	0.0	42.2	0.0	44.6
654	55	15	200	683714.63	0.0	62.3	2.1	86.3
654	79	15	400	476160.35	0.0	59.3	0.0	74.6
654	96	15	600	361544.08	0.0	46.4	0.0	56.2
654	116	15	800	304604.42	0.0	33.9	0.0	36.8
654	129	15	1000	279219.47	0.0	42.4	0.0	44.0
Averages					0.0	48.6	0.3	62.7

Table 4.9: The results of the proposed method when using optimal allocation on the instances with $n = 1060$ customers.

					Greedy allocation		Optimal allocation	
n	c	F (k)	d	Best	R. Diff. (%)	t	R. Diff. (%)	t
1060	4	1	200	974005.65	0.5	106.6	0.0	421.5
1060	9	1	400	402263.84	0.0	221.1	0.0	384.2
1060	15	1	600	368020.64	0.0	114.8	0.0	191.3
1060	23	1	800	364269.50	0.0	261.5	0.0	362.3
1060	33	1	1000	364293.23	0.0	167.9	0.0	291.5
1060	4	2	200	1941005.65	0.5	106.1	0.0	420.3
1060	9	2	400	668263.84	0.0	221.5	0.0	387.1
1060	15	2	600	530185.86	0.0	172.0	0.0	270.4
1060	23	2	800	487745.51	0.1	276.0	0.0	418.6
1060	33	2	1000	486764.67	0.0	154.0	0.0	227.3
1060	4	5	200	4842005.65	0.5	106.5	0.0	421.8
1060	9	5	400	1466263.84	0.0	221.3	0.0	379.4
1060	15	5	600	1016185.86	0.0	170.9	0.0	270.0
1060	23	5	800	764633.30	0.0	261.1	0.0	316.8
1060	33	5	1000	714854.17	0.0	140.6	0.0	212.1
1060	4	10	200	9677005.65	0.5	106.4	0.0	419.7
1060	9	10	400	2796263.84	0.0	221.7	0.0	386.5
1060	15	10	600	1826185.86	0.0	171.5	0.0	270.4
1060	23	10	800	1209980.13	0.0	266.4	0.4	318.3
1060	33	10	1000	1039858.84	0.0	183.7	0.0	282.9
1060	4	15	200	14512005.65	0.5	106.5	0.0	421.0
1060	9	15	400	4126263.84	0.0	221.5	0.0	387.1
1060	15	15	600	2636185.86	0.0	171.0	0.0	270.9
1060	23	15	800	1659980.13	0.0	267.0	0.3	317.2
1060	33	15	1000	1364858.84	0.0	183.8	0.0	280.9
Averages					0.1	184.1	0.0	333.2

5 CONCLUSIONS

The $GF|1c, d$ is an NP-hard problem that can be used for several real-world planning scenarios, such as positioning water towers, wireless network access points and electrical substations. However, modern solvers cannot provide good solutions in practical time. This creates a need for fast heuristics that provide feasible solutions for even very large instances.

The proposed algorithm is built upon the classical idea of splitting the solution of a location-allocation problem into a location-only phase and an allocation-only phase. The two phases are shown to be nearly optimal on the analyzed instances and a GAP formulation that makes both phases optimal is presented. This allows one to look at the $GF|1c, d$ as a problem of generating candidate positions for sources and a discrete facility location problem whose solution will be adjusted in continuous space.

The performance of the algorithm is comparable to the state of the art of the $GF|d$ and is vastly superior than that of a commercial solver when given a MISOCP formulation. While the solver was unable to find any solution after 10^4 seconds for instances with $n \geq 287$, the proposed method find feasible solutions for n as large as 1060 in less than 300 seconds.

For future work, a robust version of the problem could be proposed. The input data might be based on sample estimates that are not representative of the ground truth. In some cases, the position and demand of the destinations may vary over time. Because of this, being able to find solutions which are resistant to small changes in the inputs is useful.

REFERENCES

BECK, Amir; SABACH, Shoham. Weiszfeld's Method: Old and New Results. *Journal of Optimization Theory and Applications*, v. 164, p. 1–40, 2015. DOI: 10.1007/s10957-014-0586-7.

BONGARTZ, Ingrid; CALAMAI, Paul H.; CONN, Andrew R. A projection method for l_p norm location-allocation problems. *Mathematical Programming*, v. 66, n. 1, p. 283–312, Aug. 1994. ISSN 1436-4646. DOI: 10.1007/BF01581151.

BOYD, Stephen; VANDENBERGHE, Lieven. *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004. ISBN 0521833787. Available from: <https://web.stanford.edu/~boyd/cvxbook>. Visited on: 4 Sept. 2019.

BRIMBERG, J.; MLADENOVIC, N.; SALHI, S. The multi-source Weber problem with constant opening cost. *Journal of the Operational Research Society*, Taylor & Francis, v. 55, n. 6, p. 640–646, 2004. DOI: 10.1057/palgrave.jors.2601754.

BRIMBERG, J.; SALHI, S. A Continuous Location-Allocation Problem with Zone-Dependent Fixed Cost. *Annals of Operations Research*, v. 136, n. 1, p. 99–115, Apr. 2005. ISSN 1572-9338. DOI: 10.1007/s10479-005-2041-5.

BRIMBERG, Jack; HANSEN, Pierre, et al. Improvements and Comparison of Heuristics for Solving the Uncapacitated Multisource Weber Problem. *Operations Research*, INFORMS, v. 48, n. 3, p. 444–460, 2000. DOI: 10.1287/opre.48.3.444.12431.

COOPER, Leon. Heuristic Methods for Location-Allocation Problems. *SIAM Review*, Society for Industrial and Applied Mathematics, v. 6, n. 1, p. 37–53, 1964. ISSN 00361445. DOI: 10.1137/1006005.

DOUGLAS, J.; MATTHEWS, R. *Fluid Mechanics, Vol. 1*. [S.l.]: Longman, 1996.

EILON, Samuel; WATSON-GANDY, Carl Donald Tyndale; CHRISTOFIDES, Nicos. *Distribution management: mathematical modelling and practical analysis*. [S.l.]: London: Griffin, 1971. ISBN 0852641915. Available from: <http://lib.ugent.be/catalog/rug01:000475431>.

FARAHANI, Reza Zanjirani; HEKMATFAR, Masoud. *Facility Location: Concepts, Models, Algorithms and Case Studies*. [S.l.]: Springer Science & Business Media, 2009. DOI: 10.1007/978-3-7908-2151-2.

GOKBAYRAK, Kagan; KOCAMAN, Ayse Selin. A Distance-limited Continuous Location-allocation Problem for Spatial Planning of Decentralized Systems. *Comput. Oper. Res.*, Elsevier Science Ltd., Oxford, UK, UK, v. 88, n. 100, p. 15–29, Dec. 2017. ISSN 0305-0548. DOI: 10.1016/j.cor.2017.06.013.

GONG, Dijin et al. Hybrid evolutionary method for capacitated location-allocation problem. *Computers & Industrial Engineering*, v. 33, n. 3, p. 577–580, 1997. Selected Papers from the Proceedings of 1996 ICC&IC. ISSN 0360-8352. DOI: 10.1016/S0360-8352(97)00197-6.

HITCHCOCK, Frank L. The Distribution of a Product from Several Sources to Numerous Localities. *Journal of Mathematics and Physics*, v. 20, n. 1-4, p. 224–230, 1941. DOI: 10.1002/sapm1941201224. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sapm1941201224. Available from: <https://onlinelibrary.wiley.com/doi/abs/10.1002/sapm1941201224>.

KOCAMAN, Ayse Selin; HUH, Woonghee Tim; MODI, Vijay. Initial layout of power distribution systems for rural electrification: A heuristic algorithm for multilevel network design. *Applied Energy*, v. 96, p. 302–315, 2012. Smart Grids. ISSN 0306-2619. DOI: 10.1016/j.apenergy.2012.02.029.

LAPORTE, G.; NICKEL, S.; GAMA, F. S. da. *Location Science*. [S.l.]: Springer International Publishing, 2015. ISBN 9783319131115. DOI: 10.1007/978-3-319-13111-5.

LUIS, Martino; SALHI, Said; NAGY, Gábor. A Constructive Method and a Guided Hybrid GRASP for the Capacitated Multi-Source Weber Problem in the Presence of Fixed Cost. *Journal of Algorithms & Computational Technology*, v. 9, n. 2, p. 215–232, 2015. DOI: 10.1260/1748-3018.9.2.215.

________. Region-rejection based heuristics for the capacitated multi-source Weber problem. *Computers & Operations Research*, Elsevier, v. 36, n. 6, p. 2007–2017, 2009. DOI: 10.1016/j.cor.2008.06.012.

MANZOUR-AL-AJDAD, S.M.H.; TORABI, S.A.; ESHGHI, K. Single-Source Capacitated Multi-Facility Weber Problem—An iterative two phase heuristic algorithm. *Computers & Operations Research*, v. 39, n. 7, p. 1465–1476, 2012. ISSN 0305-0548. DOI: 10.1016/j.cor.2011.08.018.

ÖNCAN, Temel. Heuristics for the single source capacitated multi-facility Weber problem. *Computers & Industrial Engineering*, v. 64, n. 4, p. 959–971, 2013. ISSN 0360-8352. DOI: `10.1016/j.cie.2013.01.005`.

RAUTENBACH, Dieter et al. *Weiszfeld's algorithm revisited once again*. v. 43. [S.l.], 2004.

REINELT, Gerhard. TSPLIB—A Traveling Salesman Problem Library. *ORSA Journal on Computing*, v. 3, n. 4, p. 376–384, 1991. DOI: `10.1287/ijoc.3.4.376`.

SALHI, S. A Perturbation Heuristic for a Class of Location Problems. *The Journal of the Operational Research Society*, Palgrave Macmillan Journals, v. 48, n. 12, p. 1233–1240, 1997. ISSN 01605682, 14769360. DOI: `10.2307/3010753`.

SALHI, S.; ATKINSON, R.A. Subdrop: A modified drop heuristic for location problems. *Location Science*, v. 3, n. 4, p. 267–273, 1995. ISSN 0966-8349. DOI: `10.1016/0966-8349(96)00003-4`.

STEELE JR., Guy L.; LEA, Doug; FLOOD, Christine H. Fast Splittable Pseudorandom Number Generators. *SIGPLAN Not.*, ACM, New York, NY, USA, v. 49, n. 10, p. 453–472, Oct. 2014. ISSN 0362-1340. DOI: `10.1145/2714064.2660195`.

TODD, Michael J. *ICCOPT I Summer School: Conic Programming*. School of Operations Research and Industrial Engineering, Cornell University. 1 Aug. 2004. Available from: <`https://people.orie.cornell.edu/miketodd/iccopt.pdf`>. Visited on: 20 Aug. 2019.

WEISZFELD, E. Sur le point pour lequel la Somme des distances de n points donnés est minimum. *Tohoku Mathematical Journal, First Series*, v. 43, p. 355–386, 1937.

WEISZFELD, E.; PLASTRIA, Frank. On the point for which the sum of the distances to n given points is minimum. *Annals of Operations Research*, v. 167, n. 1, p. 7–41, Mar. 2009. ISSN 1572-9338. DOI: `10.1007/s10479-008-0352-z`.

ZAINUDDIN, Z.M.; SALHI, S. A perturbation-based heuristic for the capacitated multi-source Weber problem. *European Journal of Operational Research*, v. 179, n. 3, p. 1194–1207, 2007. ISSN 0377-2217. DOI: `10.1016/j.ejor.2005.09.050`.