mathematics

MDPI

Article

Solving the Capacitated Vertex K-Center Problem through the Minimum Capacitated Dominating Set Problem

José Alejandro Cornejo Acosta [1], Jesús García Díaz [1,2,*], Ricardo Menchaca-Méndez [3] and Rolando Menchaca-Méndez [3]

1 Instituto Nacional de Astrofísica, Óptica y Electrónica, Santa María Tonantzintla, Puebla 72840, Mexico; alexcornejo@inaoep.mx
2 Consejo Nacional de Ciencia y Tecnología, Mexico City 03940, Mexico
3 Centro de Investigación en Computación del Instituto Politécnico Nacional, Mexico City 07738, Mexico; ric@cic.ipn.mx (R.M.-M.); rmen@cic.ipn.mx (R.M.-M.)
* Correspondence: jesus.garcia@conacyt.mx

Received: 11 August 2020; Accepted: 8 September 2020; Published: 10 September 2020

Abstract: The capacitated vertex k-center problem receives as input a complete weighted graph and a set of capacity constraints. Its goal is to find a set of k centers and an assignment of vertices that does not violate the capacity constraints. Furthermore, the distance from the farthest vertex to its assigned center has to be minimized. The capacitated vertex k-center problem models real situations where a maximum number of clients must be assigned to centers and the travel time or distance from the clients to their assigned center has to be minimized. These centers might be hospitals, schools, police stations, among many others. The goal of this paper is to explicitly state how the capacitated vertex k-center problem and the minimum capacitated dominating set problem are related. We present an exact algorithm that consists of solving a series of integer programming formulations equivalent to the minimum capacitated dominating set problem over the bottleneck input graph. Lastly, we present an empirical evaluation of the proposed algorithm using off-the-shelf optimization software.

Keywords: facility location; graph theory; integer programming; optimization

1. Introduction

The capacitated vertex k-center problem is a fundamental NP-Hard problem from the family of Location Problems [1]. This problem's input is a complete weighted graph $G = (V, E)$ with edge weights that follow a metric, a capacity function $f_{cap} : V \to \mathbb{N}$, and an integer $k \in \mathbb{N}$. The goal is to find a set $C \subseteq V$ with at most k elements and an assignment function $f_C : V \setminus C \to C$. The number of vertices assigned to each vertex $v_i \in C$ must not exceed its associated capacity $f_{cap}(v_i)$, and the distance from the farthest vertex to its assigned vertex has to be minimum [2,3]. We refer to the vertices in the set C as centers.

The capacitated vertex k-center problem can be defined under different input parameters. Some authors consider each vertex $v_j \in V$ to have an associated demand $d_j \in \mathbb{N}$ [4–8]. In this case, the total demand of the vertices assigned to any center $v_i \in C$ cannot be greater than its capacity $f_{cap}(v_i)$. Some other authors consider the special case where the demand of every vertex is of one unit, and the capacity $f_{cap}(v_i)$ is the same for all vertices $v_i \in V$ [2,3]. They refer to this problem as the uniform capacitated vertex k-center problem. Other authors consider the case where demands are of one unit for all vertices, and capacities are not necessarily the same for all vertices [9]. Summarizing, by setting uniform or non-uniform demands and capacities, we obtain the main variants of the

capacitated vertex k-center problem. In this paper, we work with uniform demands and non-uniform capacities, i.e., every vertex has one unit demand, and the capacity of every vertex can be different. To avoid new terminology, we will refer to this problem just as the capacitated vertex k-center problem.

A typical application of the capacitated vertex k-center problem determines the location of a set of capacity-limited facilities where the travel time or distance from the users to the facilities has to be minimized. For instance, government agencies need to determine locations of public services like schools and hospitals so that communities can access them. Furthermore, these services may be subject to capacity limitations. In the private sector, companies must locate warehouses or distribution centers to minimize the cost of serving retail establishments from a warehouse. Usually, each warehouse is limited to serve a specific number of retail establishments. In this latter context, poor location decisions may lead to increase costs and decrease competitiveness [10]. It is important to remark that the assignment function for the capacitated vertex k-center problem is usually defined as $f_C : V \to C$ [3–9]. However, like other authors, in this paper, we define this function as $f_C : V \setminus C \to C$ [2]. This way, we can show how the capacitated vertex k-center problem can be solved through the minimum capacitated dominating set problem. Such relationship is important because both problems are usually associated with problems in different contexts. While the capacitated vertex k-center problem is mainly used to model facility location problems [1], the minimum capacitated dominating set problem is mainly used in networking problems [11,12]. Regarding the state of the art, the known exact algorithms for the capacitated vertex k-center problem are based on integer programming formulations of the capacitated concentrator location problem, the bin packing problem, and the capacitated set-covering problem [5,6,8].

The remaining of the document is organized as follows. Section 2 presents a literature review. Section 2.1 presents the classical integer programming formulation for the capacitated vertex k-center problem and Section 2.2 introduces an alternative integer programming formulation of the problem that is equivalent to the minimum capacitated dominating set problem over a bottleneck input graph. Based on this formulation, Section 3 presents an exact algorithm for solving the capacitated vertex k-center problem. It is important to point out that the goal of introducing this algorithm is explicitly stating how the capacitated vertex k-center problem is related to the minimum capacitated dominating set problem, and not necessarily to compete against the known exact algorithms from the literature. However, in Section 4, we present an empirical evaluation of the proposed algorithm that shows its usefulness over small instances using off-the-shelf optimization software. Lastly, Section 5 presents the concluding remarks.

2. The Capacitated Vertex K-Center Problem

The capacitated vertex k-center problem generalizes the more fundamental uncapacitated vertex k-center problem where any number of vertices can be assigned to each center. This way, the uncapacitated vertex k-center problem (best known as the vertex k-center problem) aims at minimizing the distance from the farthest vertex to its nearest center. Many approximation algorithms [13–18], heuristics [19,20], metaheuristics [21–25], and exact algorithms have been proposed for this problem [26–32].

The capacitated vertex k-center problem has also been approached through different algorithmic perspectives. However, the algorithms for the capacitated version have not been as successful as those for the uncapacitated version. For instance, most of the exact algorithms for the uncapacitated vertex k-center problem can solve instances with thousands of vertices. Even an instance with one million vertices has been solved [32]. Nonetheless, the exact algorithms designed for the capacitated version struggle to find optimal solutions (We refer to any element in the problem's search space as a "solution". A "feasible solution" is a solution that satisfies all the constraints of the problem. An "optimal solution" is a feasible solution of optimal size.) for instances from benchmark data sets with just some hundreds of vertices. This extra difficulty comes from the fact that in the capacitated vertex k-center problem, the output consists of a set of centers and an assignment function. Namely, this problem has location and allocation elements involved, being both an NP-Hard problem on its own [33]. Furthermore, the heuristic and metaheuristic algorithms for the capacitated version rely mostly on exploitation [4,7]. Therefore, their running time tends to become impractical as the size of the input grows. Finally, while some conceptually simple algorithms achieve the best possible approximation factor for the uncapacitated version ($\rho = 2$ under $P \neq NP$), the approximation algorithms for the capacitated versions are conceptually more complicated and have an approximation factor of 6 and 9 for the uniform-demand version with uniform and non-uniform capacities, respectively [3,9,13,14].

Among the approximation algorithms for the capacitated vertex k-center problem with uniform demands and capacities, there is a 10-approximation algorithm [2], and two 6-approximation algorithms [3,9]. To date, no one has found an algorithm with a better approximation factor for this problem. In the case of the capacitated vertex k-center problem with uniform demands and non-uniform capacities, the situation is not better; the approximation factor of the best-known approximation algorithm is 9 [9]. Since the uniform versions of the problem are a particular case of the non-uniform versions, all the algorithms designed for the non-uniform versions also get feasible solutions for the uniform versions.

Among the heuristic and metaheuristic algorithms designed for the capacitated vertex k-center problem with non-uniform demands and capacities is a large scale local search heuristic algorithm [4], a greedy randomized adaptive search procedures (GRASP) metaheuristic algorithm [34], and an iterated greedy local search and variable neighborhood descent metaheuristic algorithm [7]. Since these metaheuristics rely primarily on exploitation, their running time tends to become impractical as the input grows. However, the experimental evidence shows that these algorithms are among the most effective for this problem. Regarding exact algorithms, there are some proposals based on integer programming or mixed integer programming formulations of the problem [5,6,8].

2.1. Classical Integer Programming Formulation

Since the capacitated vertex k-center problem is a generalization of the uncapacitated vertex k-center problem, we begin by defining the latter. The uncapacitated vertex k-center problem consists in, given a complete weighted graph $G = (V, E)$, and an integer $k \in \mathbb{N}$, finding a set of centers $C \subseteq V$, $|C| \leq k$, such that the solution size $r(C)$ is minimized, where $r(C)$ (often known as covering radius) is defined as the distance from the farthest vertex in V to its nearest center in C (Equation (1)). Since the set of edge weights follows a metric, the distance between two vertices $v_i, v_j \in V$ is equal to the weight of the edge $\{v_i, v_j\} \in E$. For practicity, we may refer to the weight $w(\{v_i, v_j\})$ of each edge $\{v_i, v_j\} \in E$ as the distance between v_i and v_j. We refer to the optimal solution for the uncapacitated vertex k-center problem as C^*, which has a size of $r(C^*)$. Expressions (2) to (7) show the classical integer programming formulation for the uncapacitated vertex k-center problem [10,21]. We refer to this formulation as F1U:

$$r(C) = \max_{v \in V} d(v, C), \qquad \text{where } d(v, C) = \min_{c \in C} w(\{v, c\}) \tag{1}$$

minimize

$$z \tag{2}$$

subject to

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \forall i \in \{1, 2, ..., n\} \tag{3}$$

$$x_{ij} \leq y_j, \quad \forall i, j \in \{1, 2, ..., n\} \tag{4}$$

$$\sum_{j=1}^{n} y_j \leq k \tag{5}$$

$$\sum_{j=1}^{n} w_{ij}\, x_{ij} \leq z, \quad \forall i \in \{1, 2, ..., n\} \tag{6}$$

where

$$x_{ij}, y_j \in \{0, 1\} \tag{7}$$

In this formulation, there is a variable $y_j \in \{0,1\}$ for every vertex $v_j \in V$. If a vertex v_j is selected as a center, then $y_j = 1$; otherwise, $y_j = 0$. There is also a set of variables $x_{ij} \in \{0,1\}$. If vertex v_i is assigned to some vertex v_j that has been selected as center ($y_j = 1$), then $x_{ij} = 1$; otherwise, $x_{ij} = 0$. The value of each w_{ij} is equal to the weight of edge $\{v_i, v_j\} \in E$. This way, constraints (3) and (4) work together. While constraint (3) indicates that every vertex has to be assigned to exactly one vertex, constraint (4) indicates that the vertices can be assigned only to vertices that have been selected as centers. Constraint (5) prevents selecting more than k centers. Notice that, in the uncapacitated vertex k-center problem, the output consists only of the set of centers $C \subseteq V$, and there is no need to return an assignment function $f_C : V \setminus C \to C$. Thus, the role of performing assignments through variables x_{ij} is just to satisfy constraint (6), which assures that the distance from every vertex to its nearest center is smaller than or equal to z, which is the objective function (2) that we want to minimize. Notice that the value of z depends on the assigned vertices, which are codified through variables x_{ij}. Finally, all x_{ij} and y_j are binary variables.

By adding capacity constraints to the uncapacitated vertex k-center problem, we obtain the capacitated vertex k-center problem. This problem receives as input a complete weighted graph $G = (V, E)$, a capacity function $f_{cap} : V \to \mathbb{N}$, and an integer $k \in \mathbb{N}$. Its goal is to find a pair (C, f_C) such that $C \subseteq V$, $|C| \leq k$, $\forall v \in C, |\{(u, v) \in f_C\}| \leq f_{cap}(v)$, and the solution size $r(C, f_C)$ is minimized, where $r(C, f_C)$ is defined as the distance from the farthest vertex in $V \setminus C$ to its assigned center in C (Equation (8)) and $f_C : V \setminus C \to C$. We refer to the optimal solution for the capacitated vertex k-center problem as (C^*, f_{C^*}), which has a size of $r(C^*, f_{C^*})$. Expressions (9) to (15) show the classical integer programming formulation for the capacitated vertex k-center problem [4,5]. This formulation is referred as F1C. This formulation is similar to formulation F1U, except that it turns constraint (3) into constraint (10), and adds constraint (14):

$$r(C, f_C) = \max_{v \in V \setminus C} w(\{v, f_C(v)\}) \tag{8}$$

minimize

$$z \tag{9}$$

subject to

$$\sum_{j=1}^{n} x_{ij} = 1 - y_i, \quad \forall i \in \{1, 2, ..., n\} \tag{10}$$

$$x_{ij} \leq y_j, \quad \forall i, j \in \{1, 2, ..., n\} \tag{11}$$

$$\sum_{j=1}^{n} y_j \leq k \tag{12}$$

$$\sum_{j=1}^{n} w_{ij}\, x_{ij} \leq z, \quad \forall i \in \{1, 2, ..., n\} \tag{13}$$

$$\sum_{i=1}^{n} x_{ij} \leq f_{cap}(v_j), \quad \forall j \in \{1, 2, ..., n\} \tag{14}$$

where

$$x_{ij}, y_j \in \{0, 1\} \tag{15}$$

Since the domain of the assignment function f_C is $V \setminus C$, the vertices that are selected as centers cannot be assigned to themselves or any other center. By turning (3) into (10), this formulation prevents assigning a center to itself or any other center. Furthermore, constraint (14) indicates that the maximum number of vertices that can be assigned to every vertex v_j is at most $f_{cap}(v_j)$, where $f_{cap} : V \to \mathbb{N}$ is part of the input.

2.2. A New Formulation Based on the Minimum Capacitated Dominating Set

In addition to the classical formulations F1U and F1C, many alternative integer programming or mixed integer programming formulations have been proposed for the capacitated and uncapacitated vertex k-center problem [5,6,26–32]. Perhaps, the simplest of these formulations are based on the relationship between the uncapacitated vertex k-center problem and the NP-Hard set covering and minimum dominating set problems [26,35,36]. In fact, the uncapacitated vertex k-center problem is equivalent to the minimum dominating set problem when the size $r(C^*)$ of the optimal solution C^* is known ahead of time (Lemma 1 and Theorem 1) [35,36]. To better understand this relationship, Definition 1 describes what a dominating set is and Definition 2 describes what a minimum dominating set is. Then, Expressions (16) to (19) show the integer programming formulation for the uncapacitated vertex k-center as a minimum dominating set problem [11,36–38].

Definition 1. *Given a graph $G = (V, E)$, a dominating set is a set $D \subseteq V$ such that, for every vertex $v \in V \setminus D$, an edge $\{v, u\} \in E$ with u in D exists.*

Definition 2. *A minimum dominating set is a set of minimum cardinality among all the dominating sets.*

Definition 3. *Given a weighted graph $G = (V, E)$, a bottleneck graph $G_r = (V, E_r)$ is a graph such that E_r consists of all the edges in E with weight less than or equal to r.*

Lemma 1. *Let $G = (V, E)$ be a complete weighted graph and let k be a positive integer. If $r(C^*)$ is the size of the optimal solution C^* for the uncapacitated vertex k-center problem over $G = (V, E)$, then the minimum dominating set for the bottleneck graph $G_{r(C^*)} = (V, E_{r(C^*)})$ has at most k elements.*

Proof. The proof is by contradiction. Let D be a minimum dominating set over the bottleneck graph $G_{r(C^*)} = (V, E_{r(C^*)})$. Let us assume that $|D| > k$. Under this assumption, $\forall B \subseteq V$ such that $|B| = k$, there will always be at least one vertex $u \in V \setminus B$ that is not adjacent to some element of B. This is the same as saying that there is no edge $\{u, v\} \in E_{r(C^*)}$ such that v is in B. If such edge does not exist, then its weight must be greater than $r(C^*)$ (otherwise, it would not have been removed). Thus, the distance from vertex u to its nearest vertex in any set B must be greater than $r(C^*)$. However, this contradicts the premise that there is a set $C^* \subseteq V$ of cardinality k such that the distance from every vertex $v \in V \setminus C^*$ to its nearest vertex in C^* is less than or equal to $r(C^*)$. Therefore, the number of elements in the minimum dominating set of the bottleneck graph $G_{r(C^*)} = (V, E_{r(C^*)})$ cannot be greater than k. □

Theorem 1. *Let $G = (V, E)$ be a complete weighted graph and let k be a positive integer. If $r(C^*)$ is the size of the optimal solution C^* for the uncapacitated vertex k-center problem over $G = (V, E)$, then the optimal solution to the minimum dominating set problem over the bottleneck graph $G_{r(C^*)} = (V, E_{r(C^*)})$ is the optimal solution for the uncapacitated vertex k-center problem too.*

Proof. By Lemma 1, the minimum dominating set D of the bottleneck graph $G_{r(C^*)} = (V, E_{r(C^*)})$ has cardinality less than or equal to k. Since all edges $e \in E_{r(C^*)}$ have a weight at most $r(C^*)$, and all vertices $v \in V \setminus D$ are in the neighborhood of at least one element of D. Then, the distance from every vertex $v \in V \setminus D$ to its nearest vertex in D is also less than or equal to $r(C^*)$. Therefore, D is a set with no more than k vertices such that the distance from every vertex $v \in V \setminus D$ to its nearest vertex in D is no more than $r(C^*)$. In other words, D is the optimal solution to the uncapacitated vertex k-center problem:

minimize

$$\sum_{i=1}^{n} y_i \tag{16}$$

subject to

$$\sum_{i=1}^{n} a_{ij}\, y_i \geq 1 - y_j, \quad \forall j \in \{1, 2, ..., n\} \tag{17}$$

$$a_{ij} = \begin{cases} 1, & \text{if } w_{ij} \leq r(C^*) \text{ and } i \neq j \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

where

$$y_i \in \{0, 1\} \tag{19}$$

□

Expressions (16) to (19) show the integer programming formulation for the uncapacitated vertex k-center problem as a minimum dominating set problem. We refer to this formulation as F2U. In this formulation, there is a variable $y_i \in \{0,1\}$ for each vertex $v_i \in V$. If a vertex v_i is selected as a center, then $y_i = 1$; otherwise, $y_i = 0$. Let C be the set of selected centers. At (18), the matrix defined by all the a_{ij} values represents the adjacency matrix of the bottleneck graph $G_{r(C^*)} = (V, E_{r(C^*)})$, which results from removing all the edges with weight greater than $r(C^*)$ from the original input graph $G = (V, E)$, where $r(C^*)$ is the size of the optimal solution C^* for the uncapacitated vertex k-center problem over the input graph $G = (V, E)$. Since simple graphs do not have loops, we set a_{ij} to 0 for $i = j$. This way, constraint (17) indicates that, for every vertex $v_j \in V \setminus C$, there must be at least one edge $\{v_j, v_i\} \in E_{r(C^*)}$ such that v_i is selected as a center ($y_i = 1$), and (16) indicates that the number of selected centers must be minimized.

In more detail, constraint (17) indicates that, for each vertex v_j not selected as center ($y_j = 0$), there must be at least one vertex v_i selected as a center ($y_i = 1$) such that the edge $\{v_i, v_j\}$ is in $E_{r(C^*)}$ ($a_{ij} = 1$). This way, each vertex $v \in V \setminus C$ is adjacent to at least one center in the set of centers. Thus, by Definition 1, C is a dominating set in the bottleneck graph $G_{r(C^*)} = (V, E_{r(C^*)})$. Now, since (16) minimizes the number of elements of the set C, this set is also a minimum dominating set in the bottleneck graph $G_{r(C^*)} = (V, E_{r(C^*)})$. By Theorem 1, the optimal solution to this formulation is the optimal solution to the uncapacitated vertex k-center problem.

As well as the uncapacitated vertex k-center problem is related to the minimum dominating set problem, the capacitated vertex k-center problem is related to the minimum capacitated dominating set problem. Lemma 2 and Theorem 2 show that the capacitated vertex k-center problem is equivalent to the minimum capacitated dominating set problem when the size $r(C^*, f_{C^*})$ of the optimal solution (C^*, f_{C^*}) is known ahead of time. Here, $r(C, f_C)$ is defined as the distance from the farthest vertex to its assigned center, where $f_C : V \setminus C \to C$. Definition 4 describes what a capacitated dominating set is, and Definition 5 describes what a minimum capacitated dominating set is [11,39,40]. Expressions (22) to (28) show the proposed integer programming formulation for the capacitated vertex k-center problem as a minimum capacitated dominating set problem. We refer to this formulation as F2C.

Definition 4. *Given a graph $G = (V, E)$ and a capacity function $f_{cap} : V \to \mathbb{N}$, a capacitated dominating set $D \subseteq V$ is a set such that every vertex $v \in V \setminus D$ is assigned to some vertex $u \in D \cap N(v)$. In addition, the number of vertices assigned to each vertex $u \in D$ is not greater than its capacity $f_{cap}(u)$. Here, $N(u)$ is the neighborhood of $u \in V$.*

Definition 5. *A minimum capacitated dominating set is a set of minimum cardinality among all the capacitated dominating sets.*

Lemma 2. *Let $G = (V, E)$ be a complete weighted graph, let k be a positive integer, and let $f_{cap} : V \to \mathbb{N}$ be a capacity function. If $r(C^*, f_{C^*})$ is the size of the optimal solution (C^*, f_{C^*}) for the capacitated vertex k-center problem over $G = (V, E)$, then the optimal solution (D, f_D) for the minimum capacitated dominating set problem over the bottleneck graph $G_{r(C^*, f_{C^*})} = (V, E_{r(C^*, f_{C^*})})$ is such that D has at most k elements and f_D assigns no more than $f_{cap}(v)$ vertices to every vertex $v \in D$.*

Proof. The proof is by contradiction. Let (D, f_D) be the optimal solution to the minimum capacitated dominating set problem over the bottleneck graph $G_{r(C^*,f_{C^*})} = (V, E_{r(C^*,f_{C^*})})$. Here, $D \subseteq V$ is a dominating set, and $f_D : V \setminus D \to D$ is a function that assigns no more than $f_{cap}(v) \in \mathbb{N}$ vertices to each vertex $v \in D$. By Definition 4, $\forall (u, v) \in f_D, v \in D \cap N(u)$, and $\forall v \in D, |\{(u, v) \in f_D\}| \leq f_{cap}(v)$. Let us assume that the minimum capacitated dominating set (D, f_D) falls into some of the following cases: (a) $|D| > k$ or (b) f_D assigns more than $f_{cap}(v)$ vertices to some vertex $v \in D$. If case (a) occurs, then $\forall (B, f_B)$, where $B \subseteq V$, $|B| \leq k$, and $f_B : V \setminus B \to B$, at least one of the following statements must be true:

$$\exists (u, v) \in f_B \text{ such that } v \notin B \cap N(u) \tag{20}$$

or

$$\exists v \in B \text{ such that } |\{(u, v) \in f_B\}| > f_{cap}(v). \tag{21}$$

On one hand, if Equation (20) is true, then, there is at least one vertex $u \in V \setminus B$ that is assigned to a vertex $v \in B \setminus N(u)$. Since v is not in the neighborhood of u, $\{u, v\} \notin E_{r(C^*,f_{C^*})}$ and the distance from vertex u to v is greater than $r(C^*, f_{C^*})$; otherwise, edge $\{u, v\}$ would not have been removed from the original input graph $G = (V, E)$. This means that there is not a pair (B, f_B) of size $r(B, f_B)$ less than or equal to $r(C^*, f_{C^*})$, where $B \subseteq V$, $|B| \leq k$, and $f_B : V \setminus B \to B$ assigns no more than $f_{cap}(v)$ vertices to each vertex $v \in B$. However, our premise is that such pair exists and is the pair (C^*, f_{C^*}). Thus, Equation (20) cannot be true. On the other hand, if Equation (21) is true, then, the same pair (C^*, f_{C^*}) cannot exist. Actually, this is case (b). Therefore, case (a) and case (b) cannot be true. Namely, it cannot be true that $\forall (B, f_B)$, where $B \subseteq V$, $|B| \leq k$, and $f_B : V \setminus B \to B$, Equation (20) or Equation (21) hold. In other words, a minimum capacitated dominating set (D, f_D) over the bottleneck graph $G_{r(C^*,f_{C^*})} = (V, E_{r(C^*,f_{C^*})})$ is such that D has at most k vertices and f_D assigns no more than $f_{cap}(v)$ vertices to every vertex $v \in D$. □

Theorem 2. *Let $G = (V, E)$ be a complete weighted graph, let k be a positive integer, and let $f_{cap} : V \to \mathbb{N}$ be a capacity function. If $r(C^*, f_{C^*})$ is the size of the optimal solution (C^*, f_{C^*}) for the capacitated vertex k-center problem over $G = (V, E)$, then the optimal solution to the minimum capacitated dominating set problem over the bottleneck graph $G_{r(C^*,f_{C^*})} = (V, E_{r(C^*,f_{C^*})})$ is the optimal solution for the capacitated vertex k-center problem too.*

Proof. By Lemma 2, the optimal solution (D, f_D) to the minimum capacitated dominating set problem over the bottleneck graph $G_{r(C^*,f_{C^*})} = (V, E_{r(C^*,f_{C^*})})$ consists of a set D with no more than k elements and a function $f_D : V \setminus D \to D$ that assigns no more than $f_{cap}(v)$ vertices to every vertex $v \in D$. Since all edges $e \in E_{r(C^*,f_{C^*})}$ have weight at most $r(C^*, f_{C^*})$, and all vertices $v \in V \setminus D$ are assigned to some vertex in D. Then, the distance from every vertex $v \in V \setminus D$ to its assigned vertex in D is also less than or equal to $r(C^*, f_{C^*})$. Therefore, D is a set with no more than k vertices such that the distance from every vertex $v \in V \setminus D$ to its assigned vertex in D is no more than $r(C^*, f_{C^*})$, and no more than $f_{cap}(v)$ vertices are assigned to every vertex $v \in D$. In other words, the minimum capacitated dominating set (D, f_D) is the optimal solution to the capacitated vertex k-center problem:

minimize

$$\sum_{i=1}^{n} y_i \tag{22}$$

subject to

$$\sum_{i=1}^{n} x_{ij} \leq f_{cap}(v_j), \quad \forall j \in \{1, 2, ..., n\} \tag{23}$$

$$\sum_{i=1}^{n} x_{ji} = 1 - y_j, \quad \forall j \in \{1, 2, ..., n\} \tag{24}$$

$$x_{ij} \leq y_j, \quad \forall i, j \in \{1, 2, ..., n\} \tag{25}$$

$$x_{ij} \leq a_{ij}, \quad \forall i, j \in \{1, 2, ..., n\} \tag{26}$$

where

$$a_{ij} = \begin{cases} 1, & \text{if } w_{ij} \leq r(C^*, f_{C^*}) \text{ and } i \neq j \\ 0, & \text{otherwise} \end{cases} \tag{27}$$

$$x_{ij}, y_i \in \{0, 1\} \tag{28}$$

□

Expressions (22) to (28) show the integer programming formulation for the capacitated vertex k-center problem as a minimum capacitated dominating set problem. We refer to this formulation as F2C. In this formulation, we added variables $x_{ij} \in \{0, 1\}$, which take a value $x_{ij} = 1$ if a vertex v_i is assigned to a vertex v_j; otherwise, $x_{ij} = 0$. Constraint (23) indicates that the number of vertices assigned to any vertex $v_j \in V$ cannot be greater than its capacity $f_{cap}(v_j)$, where $f_{cap} : V \to \mathbb{N}$ is part of the input. Constraint (24) indicates that every vertex $v_j \in V \setminus C$ ($y_j = 0$) has to be assigned to exactly one vertex, and that vertices $v_j \in C$ ($y_j = 1$) does not have to be assigned to any vertex at all. Finally, constraint (25) indicates that every vertex v_i can be assigned only to vertices that have been selected as a center ($y_j = 1$) and constraint (26) indicates that every vertex v_i can be assigned only to vertices in its neighborhood ($a_{ij} = 1$). With these constraints, we add capacity restrictions and an explicit assignment to the minimum dominating set problem; i.e., a maximum of $f_{cap}(v_i)$ vertices from $V \setminus C$ must be assigned to each selected vertex $v_i \in C$ ($y_i = 1$). In addition, notice that constraints (24) to (26) guarantee that every vertex $v \in V \setminus C$ is assigned to exactly one vertex $u \in V$, which, by constraints (25) and (26), is in $C \cap N(v)$. By Definition 5, this integer programming formulation is equivalent to the minimum capacitated dominating set problem over the bottleneck graph $G_{r(C^*, f_{C^*})} = (V, E_{r(C^*, f_{C^*})})$, where $E_{r(C^*, f_{C^*})}$ contains edges from E of cost less than or equal to $r(C^*, f_{C^*})$, being $r(C^*, f_{C^*})$ the size of the optimal solution (C^*, f_{C^*}) of the capacitated vertex k-center problem over the input graph $G = (V, E)$. By Theorem 2, the optimal solution to this formulation is the optimal solution to the capacitated vertex k-center problem.

3. An Exact Algorithm for the Capacitated Vertex K-Center Problem

The integer programming formulation F2C can be solved using off-the-shelf optimization software. However, to do so, it is necessary to know the size $r(C^*, f_{C^*})$ of the optimal solution (C^*, f_{C^*}) in advance. One way to solve this issue is by using a series of guesses on the optimal solution size $r(C^*, f_{C^*})$. Since the optimal solution size $r(C^*, f_{C^*})$ must be equal to the weight of some edge in the input graph $G = (V, E)$, we can solve formulation F2C using the set of weights of the edges of the input graph. That is, replacing $r(C^*, f_{C^*})$ of Equation (27) by $w(e)$ for every $e \in E$. This will generate up to $|E|$ solutions of the form (C, f_C). Of course, not all of these solutions will be feasible. Among the obtained solutions, the optimal one will be the solution with no more than k elements, and with the minimum distance from the farthest vertex to its assigned center. Thus, this method implies that formulation F2C has to be solved $|E|$ times. Fortunately, it can be improved by performing a binary

search over the non-decreasing ordered set of edge weights of the input graph. This way, formulation F2C has to be solved only $\log |E|$ times, which is $O(\log |V|)$ times. Algorithm 1 shows the pseudocode of the proposed procedure that exploits this idea, where line 5 can be replaced by formulation F2C, setting $r(C^*, f_{C^*})$ of Equation (27) as $w(e_{mid})$. Theorem 3 shows the correctness of this algorithm.

Algorithm 1: An exact algorithm for the capacitated vertex k-center problem

Input: A complete weighted graph $G = (V, E)$, a positive integer k, a capacity function $f_{cap} : V \to \mathbb{N}$, and an ordered list of the $|E|$ edge weights of G: $w(e_1), w(e_2), ..., w(e_{|E|})$, where $w(e_i) \leq w(e_{i+1})$

Output: A set of vertices $C \subseteq V$, $|C| \leq k$, and an assignment function $f_C : V \setminus C \to C$

1 $high = |E|$;
2 $low = 1$;
3 **while** $high - low \geq 1$ **do**
4 $\quad mid = \lceil (high + low)/2 \rceil$;
5 $\quad (C, f_C) = \begin{cases} \text{Any capacitated dominating set } (D, f_D) \text{ over } G_{w(e_{mid})} & \text{, if } |D| \leq k \\ \text{A minimum capacitated dominating set } (D, f_D) \text{ over } G_{w(e_{mid})} & \text{, otherwise} \end{cases}$
6 $\quad$ **if** $mid = high$ **then**
7 $\quad\quad low = high$;
8 $\quad$ **end**
9 $\quad$ **if** $|C| \leq k$ **then**
10 $\quad\quad high = mid$;
11 $\quad$ **else**
12 $\quad\quad low = mid$;
13 $\quad$ **end**
14 **end**
15 **return** (C, f_C) ;

Theorem 3. *Algorithm 1 returns an optimal solution for the capacitated vertex k-center problem.*

Proof. The goal of Algorithm 1 is to use some off-the-shelf optimization software to solve the integer programming formulation F2C for the capacitated vertex k-center problem (line 5) over the input graph G with a value of $w(e_{mid})$ that equals the optimal solution size $r(C^*, f_{C^*})$, where (C^*, f_{C^*}) is the optimal solution to the capacitated vertex k-center problem. To achieve this, Algorithm 1 performs a binary search (*while* loop from line 3 to 14) over the non-decreasing ordered set of edge weights of the input graph. Every time a solution (C, f_C) with $|C| \leq k$ is generated, *high* is set to *mid* (line 10); otherwise, *low* is set to *mid* (line 12). This is because, when the minimum capacitated dominating set of a bottleneck graph G' has more than k elements, it is necessary to add more edges to this graph in order to get a minimum capacitated dominating set with fewer elements, and setting *low* to *mid* implies that the next value of $w(e_{mid})$ will be greater than the previous one. Therefore, it will generate a new bottleneck graph with more edges, including the edges of the previous bottleneck graph. Now, if the minimum capacitated dominating set of a bottleneck graph G' has k or fewer elements; then, it is possible that by removing more edges, we can still find a minimum capacitated dominating set with k or fewer elements. Thus, setting *high* to *mid* implies that the next value of $w(e_{mid})$ is going to be smaller. Therefore, it will generate a new bottleneck graph with fewer edges. In this manner, at the last iteration of the *while* loop, the value of $w(e_{mid})$ is equal to $w(e_{high})$, which implies that the bottleneck graph generated by $w(e_{mid})$ has a minimum capacitated dominating set with k or less elements. Furthermore, $w(e_{low})$ equals $w(e_{mid-1})$ and any bottleneck graph generated by any number less than or equal to

$w(e_{low})$ has a minimum capacitated dominating set with more than k elements. Therefore, at the last iteration of the *while* loop, the value of $w(e_{mid})$ is the smallest one from the set of edge weights in E that is capable of generating a bottleneck graph with a capacitated dominating set with at most k elements, i.e., $w(e_{mid}) = r(C^*, f_{C^*})$. Thus, Algorithm 1 solves the capacitated vertex k-center problem optimally. Finally, the *if* condition from lines 6 to 8 assures that the stop condition of the binary search is met. □

4. Empirical Performance Evaluation

To test Algorithm 1, two sets of instances with non-uniform capacities were created. The instances from the first set have 100 to 107 vertices (Set 1) and the instances from the second set have 200 to 280 vertices (Set 2). All instances were created as follows. A benchmark instance from TSPLib [41] was selected. Then, from this instance, four variants with non-uniform capacities were generated. For example, from instance kroA100, instances kroA100_Q1 to kroA100_Q4 were created and the capacity associated with each vertex was selected at random from the range $[\lfloor 0.75 \cdot \frac{n-k}{k} \rfloor, \lceil 1.25 \cdot \frac{n-k}{k} \rceil]$, where n is the number of vertices and k is the number of centers. This range was used because $\lceil \frac{n-k}{k} \rceil$ is the minimum capacity that every vertex must have to guarantee that all vertices can be assigned to a center when capacities are uniform. Since centers are not assigned, k is subtracted from the number of vertices n in the numerator. This way, by selecting random capacities from the range $[\lfloor 0.75 \cdot \frac{n-k}{k} \rfloor, \lceil 1.25 \cdot \frac{n-k}{k} \rceil]$, the capacity of each vertex will be relatively close to $\lceil \frac{n-k}{k} \rceil$. This is important because, as the capacity of all vertices approaches n, the problem becomes closer to the uncapacitated version, which is easier to solve according to empirical experiments [32,36]. Thus, the generated instances remain relatively difficult to solve. Regarding the value of k, it was selected as follows. For $Q1$ instances, $k = 5$; for $Q2$ instances, $k = 10$; for $Q3$ instances, $k = 20$; and for $Q4$ instances, $k = 40$.

Tables 1 and 2 show the results obtained by the proposed exact algorithm (Algorithm 1) and by the classical formulation F1C over Set 1 and Set 2, respectively. The experiments were performed using Gurobi version 9.0.0 as off-the-shelf optimization software with its default tuning parameters [42]. Regarding Algorithm 1, we implemented line 5 by setting Gurobi to stop as soon as a feasible solution of size less than or equal to k is found. This way, Algorithm 1 can reduce its execution time. All the experiments were performed on an Asus laptop with an Intel Core i5-8300H processor (Santa Clara, CA, USA) and 16 GB of RAM. The set of instances and the source code of the algorithms can be obtained from https://github.com/alex-cornejo/exact_ckc. From Table 1, we can observe that Algorithm 1 needs an average of 9.61 s to solve each instance, while the classical formulation F1C needs an average of 30.58 s. Namely, Algorithm 1 is at least three times faster. Furthermore, Algorithm 1 converged faster to the optimal solution in 37 out of the 48 instances. From Table 2, we can observe that Algorithm 1 needs an average of 1332 s to solve each instance, while the classical formulation F1C needs an average time greater than 13,726 s. Thus, for Set 2, Algorithm 1 is at least 10 times faster. Actually, we had to set a maximum execution time of 24 h for the classical formulation F1C because this time was not enough for the classical formulation to solve some instances. Furthermore, Algorithm 1 converged faster to the optimal solution in 39 out of the 48 instances. From these results, we can observe that the tested algorithms are better suited for solving relatively small instances of the problem and that the proposed Algorithm 1 tends to converge faster to the optimal solution.

Table 1. Running time and optimal solution size found by Algorithm 1 and the classical formulation F1C over Set 1 using Gurobi 9.0.0. Best running times are in bold.

				Running Time (Seconds)	
Instance	***n***	***k***	**OPT**	**Algorithm 1**	**F1C**
kroA100_Q1	100	5	895.64	**2.32**	10.42
		6	814.87	**1.46**	4.18
kroA100_Q2	100	10	606.57	**1.86**	12.64
		11	554.04	**1.43**	9.07
kroA100_Q3	100	20	411.61	**4.95**	32.45
		21	376.96	**2.65**	33.78
kroA100_Q4	100	40	325.04	**1.00**	12.90
		41	314.23	**0.72**	2.67
kroB100_Q1	100	5	924.57	**3.39**	9.27
		6	823.66	**1.23**	5.43
kroB100_Q2	100	10	602.9	**2.25**	7.44
		11	559.35	**1.34**	4.99
kroB100_Q3	100	20	425.73	**11.91**	26.30
		21	414.77	**6.56**	19.76
kroB100_Q4	100	40	343.57	**0.77**	2.22
		41	330.84	**0.75**	2.69
kroC100_Q1	100	5	867.16	**1.79**	6.61
		6	762.27	**1.07**	8.94
kroC100_Q2	100	10	580.53	**2.69**	3.68
		11	545.69	**2.27**	3.32
kroC100_Q3	100	20	426.82	**29.29**	38.61
		21	415.32	**13.89**	124.56
kroC100_Q4	100	40	307.65	**0.90**	2.50
		41	288.53	**0.65**	1.39
eil101_Q1	101	5	21.21	**4.44**	10.40
		6	18.68	**1.62**	9.03
eil101_Q2	101	10	15.23	**17.63**	25.27
		11	13.92	**6.96**	18.20
eil101_Q3	101	20	10.44	10.66	**9.04**
		21	10.29	7.51	**5.15**
eil101_Q4	101	40	8.6	4.94	**1.59**
		41	8.48	2.32	**1.89**
lin105_Q1	105	5	677.44	**6.14**	7.31
		6	610.45	**2.81**	6.69
lin105_Q2	105	10	555.0	99.23	**27.35**
		11	476.05	152.56	**27.1**
lin105_Q3	105	20	307.0	**3.51**	15.74
		21	307.0	**3.35**	25.69
lin105_Q4	105	40	177.01	**1.31**	2.09
		41	162.69	**0.84**	1.62
pr107_Q1	107	5	2630.58	**21.13**	868.16
		6	1068.87	**1.58**	2.31
pr107_Q2	107	10	894.42	**3.07**	8.51
		11	824.62	2.79	**2.75**
pr107_Q3	107	20	538.51	3.17	**1.55**
		21	447.21	3.04	**2.48**
pr107_Q4	107	40	282.84	1.88	**1.10**
		41	282.84	1.86	**0.93**
Average time				**9.61**	30.58

Table 2. Running time and optimal solution size found by Algorithm 1 and the classical formulation F1C over Set 2 using Gurobi 9.0.0. Best running times are in bold.

				Running Time (Seconds)	
Instance	***n***	***k***	**OPT**	**Algorithm 1**	**F1C**
kroa200_Q1	200	5	919.03	**25.61**	133.75
		6	808.66	**10.53**	67.71
kroa200_Q2	200	10	599.47	**34.48**	247.57
		11	569.94	**41.1**	443.19
kroa200_Q3	200	20	415.49	**534.92**	1568.01
		21	403.1	**824.89**	852.34
kroa200_Q4	200	40	293.29	**76.65**	242.6
		41	287.45	**59.69**	166.92
kroB200_Q1	200	5	897.66	**17.59**	83.57
		6	784.18	**9.22**	52.55
kroB200_Q2	200	10	589.86	**41.29**	904.5
		11	567.5	**27.26**	126.71
kroB200_Q3	200	20	412.14	**1352.51**	3903.28
		21	399.48	**391.72**	1230.63
kroB200_Q4	200	40	289.27	**483.67**	37,907.53
		41	282.4	**69.25**	35,200.62
ts225_Q1	225	5	4000.0	127.16	**118.97**
		6	3605.55	**44.26**	108.12
ts225_Q2	225	10	3041.38	**840.59**	3037.34
		11	3000.0	**176.57**	4372.96
ts225_Q3	225	20	2000.0	**407.86**	473.29
		21	1802.77	**224.0**	1566.91
ts225_Q4	225	40	1414.21	**194.32**	1115.63
		41	1118.03	**170.12**	4623.0
pr226_Q1	226	5	4172.52	**140.19**	481.23
		6	3778.97	**53.08**	213.13
pr226_Q2	226	10	2863.56	**102.9**	>86,400
		11	2844.29	**73.39**	>86,400
pr226_Q3	226	20	2450.51	**410.74**	>86,400
		21	2300.54	**186.73**	>86,400
pr226_Q4	226	40	1320.98	**162.73**	>86,400
		41	1166.19	**87.73**	>86,400
gr229_Q1	229	5	50.26	15971.84	**3036.74**
		6	37.94	**120.27**	192.09
gr229_Q2	229	10	37.94	9548.37	**746.46**
		11	28.84	4216.9	**685.63**
gr229_Q3	229	20	23.23	8412.5	**1129.62**
		21	22.61	2091.64	**688.22**
gr229_Q4	229	40	19.78	4738.61	**1135.16**
		41	19.67	7827.95	**745.42**
a280_Q1	280	5	69.85	734.93	**626.56**
		6	58.24	**50.37**	340.37
a280_Q2	280	10	45.25	**93.96**	333.76
		11	42.52	**66.67**	328.59
a280_Q3	280	20	31.24	**700.75**	6289.73
		21	28.84	**376.45**	2186.42
a280_Q4	280	40	21.54	**846.97**	1046.82
		41	20.39	**772.48**	21,710.55
Average time				**1332.78**	>13,726.34

5. Conclusions and Future Work

This paper shows that the capacitated vertex k-center problem can be solved through the minimum capacitated dominating set problem. This is accomplished by introducing Algorithm 1, which consists of solving $O(\log |V|)$ times an integer programming formulation that is equivalent to the NP-Hard minimum capacitated dominating set problem. Additionally, this paper presents some experiments showing the usefulness of the proposed algorithm for solving small instances of the problem.

Since the capacitated vertex k-center problem is an NP-Hard problem, no algorithm can solve it in polynomial time (under $P \neq NP$). Thus, as well as other exact algorithms proposed for this problem, Algorithm 1 is an exponential-time algorithm limited to solving small instances. However, this algorithm can be used as a basis for designing efficient heuristics or approximation algorithms. These algorithms may find near-optimal feasible solutions for arbitrary instances in polynomial time. Finally, our proposal can also serve as a basis for designing exact algorithms for related problems, such as the capacitated vertex k-center problem with non-uniform demands.

Author Contributions: Conceptualization, J.A.C.A., J.G.D., R.M.-M. (Ricardo Menchaca-Méndez), and R.M.-M. (Rolando Menchaca-Méndez); Formal analysis, J.A.C.A., J.G.D., R.M.-M. (Ricardo Menchaca-Méndez), and R.M.-M. (Rolando Menchaca-Méndez); Investigation, J.A.C.A. and J.G.D.; Methodology, J.A.C.A., J.G.D., R.M.-M. (Ricardo Menchaca-Méndez), and R.M.-M. (Rolando Menchaca-Méndez); Software, J.A.C.A. and J.G.D.; Validation, R.M.-M. (Ricardo Menchaca-Méndez) and R.M.-M. (Rolando Menchaca-Méndez); Writing—original draft, J.A.C.A. and J.G.D.; Writing—review and editing, J.A.C.A., J.G.D., R.M.-M. (Ricardo Menchaca-Méndez), and R.M.-M. (Rolando Menchaca-Méndez). All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: We acknowledge Gurobi for providing a free-of-charge academic license for Gurobi version 9.0.0, which was used in the computations presented in this paper. The authors are grateful to anonymous reviewers whose questions, comments, and suggestions helped improve an earlier version of this paper. The authors also acknowledge Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE) and Consejo Nacional de Ciencia y Tecnología (CONACYT) for providing the necessary resources for the development of this research.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Laporte, G.; Nickel, S.; da Gama, F.S. *Location Science*; Springer International Publishing: Berlin/Heidelberg, Germany, 2019.
2. Barilan, J.; Kortsarz, G.; Peleg, D. How to allocate network centers. *J. Algorithms* **1993**, *15*, 385–415. [CrossRef]
3. Khuller, S.; Sussmann, Y.J. The capacitated k-center problem. *SIAM J. Discret. Math.* **2000**, *13*, 403–418. [CrossRef]
4. Scaparra, M.P.; Pallottino, S.; Scutellà, M.G. Large-scale local search heuristics for the capacitated vertex p-center problem. *Netw. Int. J.* **2004**, *43*, 241–255.
5. Özsoy, F.A.; Pınar, M.Ç. An exact algorithm for the capacitated vertex p-center problem. *Comput. Oper. Res.* **2006**, *33*, 1420–1436. [CrossRef]
6. Albareda-Sambola, M.; Díaz, J.A.; Fernández, E. Lagrangean duals and exact solution to the capacitated p-center problem. *Eur. J. Oper. Res.* **2010**, *201*, 71–81. [CrossRef]
7. Quevedo-Orozco, D.R.; Ríos-Mercado, R.Z. Improving the quality of heuristic solutions for the capacitated vertex p-center problem through iterated greedy local search with variable neighborhood descent. *Comput. Oper. Res.* **2015**, *62*, 133–144. [CrossRef]
8. Kramer, R.; Iori, M.; Vidal, T. Mathematical models and search algorithms for the capacitated p-center problem. *INFORMS J. Comput.* **2019**, *32*, 444–460. [CrossRef]
9. An, H.C.; Bhaskara, A.; Chekuri, C.; Gupta, S.; Madan, V.; Svensson, O. Centrality of trees for capacitated k-center. *Math. Program.* **2015**, *154*, 29–53. [CrossRef]
10. Daskin, M.S. *Network and Discrete Location: Models, Algorithms, and Applications*; John Wiley & Sons: Hoboken, NJ, USA, 2011.
11. Potluri, A.; Singh, A. Metaheuristic algorithms for computing capacitated dominating set with uniform and variable capacities. *Swarm Evol. Comput.* **2013**, *13*, 22–33. [CrossRef]

12. Yuan, F.; Li, C.; Gao, X.; Yin, M.; Wang, Y. A novel hybrid algorithm for minimum total dominating set problem. *Mathematics* **2019**, *7*, 222. [CrossRef]
13. Hochbaum, D.S.; Shmoys, D.B. A best possible heuristic for the k-center problem. *Math. Oper. Res.* **1985**, *10*, 180–184. [CrossRef]
14. Gonzalez, T.F. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.* **1985**, *38*, 293–306. [CrossRef]
15. Dyer, M.E.; Frieze, A.M. A simple heuristic for the p-centre problem. *Oper. Res. Lett.* **1985**, *3*, 285–288. [CrossRef]
16. Plesník, J. A heuristic for the p-center problems in graphs. *Discrete Appl. Math.* **1987**, *17*, 263–268. [CrossRef]
17. Shmoys, D.B. Computing near-optimal solutions to combinatorial optimization problems. *Comb. Optim.* **1995**, *20*, 355–397.
18. Garcia-Diaz, J.; Sanchez-Hernandez, J.; Menchaca-Mendez, R.; Menchaca-Mendez, R. When a worse approximation factor gives better performance: A 3-approximation algorithm for the vertex k-center problem. *J. Heuristics* **2017**, *23*, 349–366. [CrossRef]
19. Rana, R.; Garg, D. The analytical study of k-center problem solving techniques. *Int. J. Inf. Technol. Knowl. Manag.* **2008**, *1*, 527–535.
20. Robič, B.; Mihelič, J. Solving the k-center problem efficiently with a dominating set algorithm. *J. Comput. Inf. Technol.* **2005**, *13*, 225–234.
21. Mladenović, N.; Labbé, M.; Hansen, P. Solving the p-center problem with tabu search and variable neighborhood search. *Netw. Int. J.* **2003**, *42*, 48–64. [CrossRef]
22. Pacheco, J.A.; Casado, S. Solving two location models with few facilities by using a hybrid heuristic: A real health resources case. *Comput. Oper. Res.* **2005**, *32*, 3075–3091. [CrossRef]
23. Pullan, W. A memetic genetic algorithm for the vertex p-center problem. *Evol. Comput.* **2008**, *16*, 417–436. [CrossRef] [PubMed]
24. Davidović, T.; Ramljak, D.; Šelmić, M.; Teodorović, D. Bee colony optimization for the p-center problem. *Comput. Oper. Res.* **2011**, *38*, 1367–1376. [CrossRef]
25. Kaveh, A.; Nasr, H. Solving the conditional and unconditional p-center problem with modified harmony search: A real case study. *Sci. Iran.* **2011**, *18*, 867–877. [CrossRef]
26. Daskin, M.S. A new approach to solving the vertex p-center problem to optimality: Algorithm and computational results. *Commun. Oper. Res. Soc. Jpn.* **2000**, *45*, 428–436.
27. Ilhan, T.; Ozsoy, F.; Pinar, M. *An Efficient Exact Algorithm for the Vertex P-Center Problem and Computational Experiments for Different Set Covering Subproblems*; Technical Report; Department of Industrial Engineering, Bilkent University: Ankara, Turkey, 2002.
28. Elloumi, S.; Labbé, M.; Pochet, Y. A new formulation and resolution method for the p-center problem. *INFORMS J. Comput.* **2004**, *16*, 84–94. [CrossRef]
29. Al-Khedhairi, A.; Salhi, S. Enhancements to two exact algorithms for solving the vertex p-center problem. *J. Math. Model. Algorithms* **2005**, *4*, 129–147. [CrossRef]
30. Chen, D.; Chen, R. New relaxation-based algorithms for the optimal solution of the continuous and discrete p-center problems. *Comput. Oper. Res.* **2009**, *36*, 1646–1655. [CrossRef]
31. Calik, H.; Tansel, B.C. Double bound method for solving the p-center location problem. *Comput. Oper. Res.* **2013**, *40*, 2991–2999. [CrossRef]
32. Contardo, C.; Iori, M.; Kramer, R. A scalable exact algorithm for the vertex p-center problem. *Comput. Oper. Res.* **2019**, *103*, 211–220. [CrossRef]
33. Marinakis, Y., Location Routing Problem. In *Encyclopedia of Optimization*; Floudas, C.A., Pardalos, P.M., Eds.; Springer: Boston, MA, USA, 2009; pp. 1919–1925. [CrossRef]
34. Quevedo-Orozco, D.R.; Ríos-Mercado, R.Z. A new heuristic for the capacitated vertex p-center problem. *Conference of the Spanish Association for Artificial Intelligence*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 279–288.
35. Minieka, E. The m-center problem. *Siam Rev.* **1970**, *12*, 138–139. [CrossRef]
36. Garcia-Diaz, J.; Menchaca-Mendez, R.; Menchaca-Mendez, R.; Hernández, S.P.; Pérez-Sansalvador, J.C.; Lakouari, N. Approximation Algorithms for the Vertex K-Center Problem: Survey and Experimental Evaluation. *IEEE Access* **2019**, *7*, 109228–109245. [CrossRef]

37. Li, R.; Hu, S.; Liu, H.; Li, R.; Ouyang, D.; Yin, M. Multi-Start Local Search Algorithm for the Minimum Connected Dominating Set Problems. *Mathematics* **2019**, *7*, 1173. [CrossRef]
38. Cabrera Martínez, A.; Hernández-Gómez, J.C.; Inza, E.P.; Sigarreta, J.M. On the Total Outer k-Independent Domination Number of Graphs. *Mathematics* **2020**, *8*, 194. [CrossRef]
39. Liedloff, M.; Todinca, I.; Villanger, Y. Solving capacitated dominating set by using covering by subsets and maximum matching. *Discret. Appl. Math.* **2014**, *168*, 60–68. [CrossRef]
40. Li, R.; Hu, S.; Zhao, P.; Zhou, Y.; Yin, M. A novel local search algorithm for the minimum capacitated dominating set. *J. Oper. Res. Soc.* **2018**, *69*, 849–863. [CrossRef]
41. Reinelt, G. TSPLIB—A traveling salesman problem library. *ORSA J. Comput.* **1991**, *3*, 376–384. [CrossRef]
42. Gurobi Optimization. Gurobi Optimizer Reference Manual. 2020. Available online: https://www.gurobi.com/documentation/9.0/refman (accessed on 11 July 2020).