EXACT SOLUTION METHODOLOGIES FOR THE P-CENTER PROBLEM UNDER SINGLE AND MULTIPLE ALLOCATION STRATEGIES

A DISSERTATION SUBMITTED TO

THE DEPARTMENT OF INDUSTRIAL ENGINEERING

AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

By

Hatice Çalık

December, 2013

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Assoc. Prof. Dr. Oya Karaşan(Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Assoc. Prof. Dr. Bahar Yetiş Kara(Co-Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Prof. Dr. M. Selim Aktürk

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Asst. Prof. Dr. Sibel Alumur Alev

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Assoc. Prof. Dr. İbrahim Körpeoğlu

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Assoc. Prof. Dr. Hande Yaman

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Graduate School

ABSTRACT

EXACT SOLUTION METHODOLOGIES FOR THE P-CENTER PROBLEM UNDER SINGLE AND MULTIPLE ALLOCATION STRATEGIES

Hatice Çalık
Ph.D. in Industrial Engineering
Supervisor: Assoc. Prof. Dr. Oya Karaşan
Co-supervisor: Assoc. Prof. Dr. Bahar Yetiş Kara
December, 2013

The p-center problem is a relatively well known facility location problem that involves locating p identical facilities on a network to minimize the maximum distance between demand nodes and their closest facilities. The focus of the problem is on the minimization of the worst case service time. This sort of objective is more meaningful than total cost objectives for problems with a time sensitive service structure. A majority of applications arises in emergency service locations such as determining optimal locations of ambulances, fire stations and police stations where the human life is at stake. There is also an increased interest in p-center location and related location covering problems in the contexts of terror fighting, natural disasters and human-caused disasters. The p-center problem is NP-hard even if the network is planar with unit vertex weights, unit edge lengths and with the maximum vertex degree of 3. If the locations of the facilities are restricted to the vertices of the network, the problem is called the vertex restricted p-center problem; if the facilities can be placed anywhere on the network, it is called the absolute p-center problem. The p-center problem with capacity restrictions on the facilities is referred to as the capacitated p-center problem and in this problem, the demand nodes can be assigned to facilities with single or multiple allocation strategies. In this thesis, the capacitated p-center problem under the multiple allocation strategy is studied for the first time in the literature.

The main focus of this thesis is a modelling and algorithmic perspective in the exact solution of absolute, vertex restricted and capacitated p-center problems. The existing literature is enhanced by the development of mathematical formulations that can solve typical dimensions through the use of off the-shelf

commercial solvers. By using the structural properties of the proposed formulations, exact algorithms are developed. In order to increase the efficiency of the proposed formulations and algorithms in solving higher dimensional problems, new lower and upper bounds are provided and these bounds are utilized during the experimental studies. The dimensions of problems solved in this thesis are the highest reported in the literature.

Keywords: p-center problem, absolute p-center problem, capacitated p-center problem, multiple allocation, branch and cut algorithm, Benders Decomposition, network design.

ÖZET

TEKLİ VE ÇOKLU ATAMA STRATEJİLERİ ALTINDA P-MERKEZ PROBLEMİ İÇİN KESİN ÇÖZÜM YÖNTEMLERİ

Hatice Çalık
Endüstri Mühendisliği, Doktora
Tez Yöneticisi: Doç. Dr. Oya Karaşan
Eş Tez Yöneticisi: Doç. Dr. Bahar Yetiş Kara
Aralık, 2013

Literatürde yaygın olarak bilinen p-merkez problemi, verilen bir serim üzerine p adet merkez yerleştirilmesini, serimdeki talep noktalarının bu merkezlerden hizmet alacak şekilde atamalarının yapılmasını ve bu atama mesafelerinin en büyüğünün en küçüklenmesini amaçlar. Problem düzlemsel serimlerde dahi NP-Zor sınıfında bir problemdir. Ancak ağaç serimlerde polinom zamanlı algoritmalar mevcuttur. Problemin temel uygulama alanlarından bazıları ambulans, itfaiye gibi acil servis ünitelerinin yerleştirilmesi ve doğal afet sonrası arama kurtarma ekiplerinin afet bölgelerine atanmasıdır. Problem aday merkezlerin kümesine göre ikiye ayrılır. Serim üzerindeki her noktanın bir aday merkez olduğu problem mutlak p-merkez problemi, sadece düğümlerin aday merkez olabildiği problem ise düğüm kısıtlı p-merkez problemi olarak adlandırılır. Merkezlerin hizmet kapasitesinin sınırlı olduğu problemlere kapasite kısıtlı p-merkez problemi adı verilir. Kapasite kısıtlı p-merkez probleminde talep noktalarının merkezlere atanmasında tekli ya da çoklu atama stratejileri güdülebilir. Çoklu atama stratejisinin güdüldüğü kapasite kısıtlı p-merkez problemi üzerine ilk kez bu tezde odaklanılmış ve bu problemin çözümüne yönelik yeni algoritmalar geliştirilmiştir.

Bu tezde mutlak ve düğüm kısıtlı p-merkez problemi ile tekli ve çoklu atama stratejileri altındaki kapasite kısıtlı p-merkez problemlerinin çözümüne yönelik modelleme ve algoritma tabanlı yöntemler geliştirilmesi amaçlanmıştır. Bu doğrultuda öncelikle bu problemler üzerine literatürde yapılan çalışmaların geniş çaplı taraması yapılmıştır. Bu problemler için öncelikli olarak çeşitli matematiksel modeller oluşturulmuş, bu modellerin yapısal özellikleri kullanılarak optimal çözüm veren algoritmalar geliştirilmiştir. Bu tezde geliştirilen algoritmaları ve matematiksel modelleri daha hızlı çözebilmek amacıyla, yeni alt ve üst

sınırlar elde edilmesini sağlayan yöntemler sunulmuş, bu alt ve üst sınırlar kullanılarak geniş ölçekli problemler üzerinde testler gerçekleştirilmiş ve daha önce çözülemeyen büyüklükteki problemler çözülmüştür.

Anahtar sözcükler: p-merkez problemi, mutlak p-merkez problemi, kapasite kısıtlı p-merkez problemi, çoklu atama, dal kesi algoritması, Benders çözünürlük yöntemi, ağ tasarımı.

To my dear Professor Barbaros Tansel...

Acknowledgement

First and foremost, I would like to express my deep and sincere gratitude to my supervisors Assoc. Prof. Oya Karaşan and Assoc. Prof. Bahar Yetiş Kara. They were magnificent mentors and it was a great pleasure to work with them. They have always been supporting and encouraging during my PhD studies and career decisions. I would like to thank both of them for their everlasting patience and invaluable guidance.

I keep my special thanks for Assoc. Prof. Hande Yaman and Asst. Prof. Sibel Alumur Alev. They read meticulously each part of my work and provided precious suggestions and support. Honestly, I feel very lucky to have such a great thesis committee.

I am also grateful to Prof. Selim Aktürk and Assoc. Prof. İbrahim Körpeoğlu for accepting to be a member of my examination committee and devoting their valuable time for reading this thesis. Their suggestions and comments are of great value to the quality of this thesis. I also want to thank to Assoc. Prof. Alper Şen and Asst. Prof. Ayşegül Altın Kayhan for accepting to be the additional members of my examination committee.

It was a great pleasure to meet each member of the Industrial Engineering Department of Bilkent University. I want to send my special thanks to Yeşim Karadeniz, Prof. Ülkü Gürler, Prof. Nesim Erkip, Asst. Prof. Yiğit Karpat, Asst. Prof. Kağan Gökbayrak, Assoc. Prof. Osman Oğuz, Figen Eren Vardal and Prof. İhsan Sabuncuoğlu for their moral support during my PhD studies.

My warm thanks go to the great friends I met in Bilkent University during my graduate study. I would like to thank to Ece Demirci, with whom I shared not only my office but also my happiness and sorrow for more than five years, my housemate Dr. Başak Renklioğlu for her everlasting support and invaluable friendship, Esra Koca and Burak Paç for being such wonderful friends, Gizem Özbaygın for her special friendship and for the Turkish coffee sessions, Ramez Kian and Sinan Bayraktar for being such amazing colleagues. I also thank to Başak Yazar,

Bengisu Sert, Bilgesu Çetinkaya, Dilek Keyf, Halenur Şahin, Hüseyin Gürkan, Irfan Mahmutoğulları, Meltem Peker, Merve Meraklı, Murat Tiniç, Nihal Berktaş, Oğuz Çetin, Okan Dükkancı, Özge Şafak, and Özüm Korkmaz. I would like to thank to also my special friends Pelin Damcı Kurt, Mehmet Can Kurt, Gökçe Akın Aras, Korhan Aras, Can Öz, Gülşah Hançerlioğulları, Fatma Sütcü, Neslihan Arslanbaba and another part of the thanks is due to Barış Cem Şal, Vedat Bayram, Okan Arslan, Barış Yıldız, Haluk Eliş, Utku Koç, Yahya Saleh, Fırat Kılcı, Feyza Güliz Şahinyazan, and Görkem Özdemir.

My mother Suna Çalık and my father Mustafa Çalık deserve my deepest gratitude. I am grateful to them for their love, support, and trust at all stages of my life. I also thank to my sisters Zeliha, Meral and my brother Servet.

I would like to acknowledge that this research was supported by grant 111M520 of Program 1001 of TÜBİTAK, The Scientific and Technological Research Council of Turkey.

I dedicate my thesis to Prof. Barbaros Tansel who was my PhD advisor until the very beginning of 2013. I am indebted to him for his magnificent guidance and encouragement. I feel very lucky and privileged to work under his supervision. May his soul rest in peace.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The p-center problem is a well known facility location problem that involves locating p identical facilities on a network to minimize the maximum distance between demand nodes and their closest facilities. The main concern of this problem is to keep the worst case service level as high as possible. This sort of objective is more meaningful than total cost objectives for problems with a time sensitive service structure. A majority of applications arises in emergency service locations such as determining optimal locations of ambulances, fire stations, and police stations where the human life is at stake. In these problems, the service level is higher if the time spent on the way in providing service (which is generally proportional to the distance traveled) is lower.

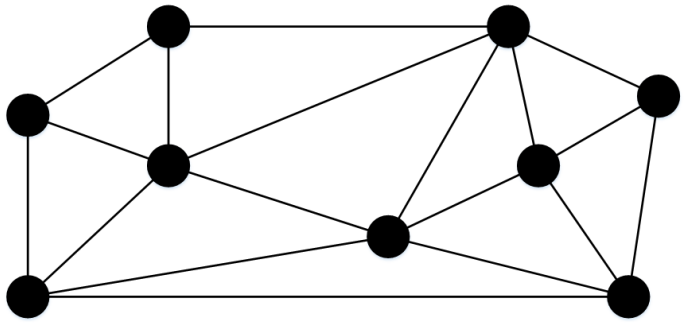
There is also an increased interest in p-center location and related location covering problems in the contexts of terror fighting, natural disasters and human-caused disasters. During an earthquake or float, the time for individuals to stay alive is restricted due to lack of clean water and food or injury. Therefore, minimization of the worst case service time plays a key role in planning of evacuation and rescue services. Another recent application of the p-center problem is in the context of evacuation from buildings and location of safe rooms.

The p-center problem can be applied to non-emergency service systems as well. One example is the location of family physicians. In many countries, individuals

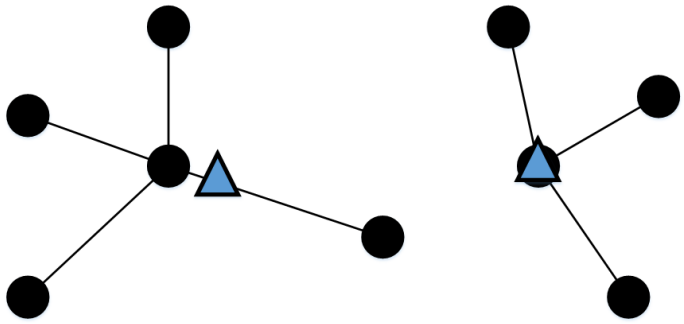
are assigned to certain family physicians and they are expected to consult primarily to their physicians. The location and allocation phase of this system can be arranged by solving a p-center problem since everybody wants to be close to his/her doctor. Other examples would be the location of public service facilities such as bank offices, libraries, school bus stops, and public school districting.

The p-center problem can be classified into two categories as absolute and vertex restricted according to the placement of the facilities over the physical infrastructure that is considered as a network. In the absolute p-center problem the facilities can be placed on vertices (nodes) or anywhere on the edges while in the vertex restricted p-center problem the facilities have to be placed on the vertices of the network. The restriction in the latter problem might be due to unavailability or nonconformity of the service on the edges of the network or it might be a managerial choice depending on the nature of the underlying real life problem. We provide an illustration of the sample optimal solutions of the absolute and vertex restricted p-center problems for $p = 2$ on a Euclidean network in Figure 1.1. In this figure, circles represent the demand nodes and triangles represent the facilities. In some applications of the p-center problem, the vertices of the network might have non-identical weight values. These weight values might correspond to a priority criterion or a constant factor to multiply by the distance for obtaining the travel time. This problem is referred to as the weighted p-center problem or the p-center problem on (vertex) weighted networks.

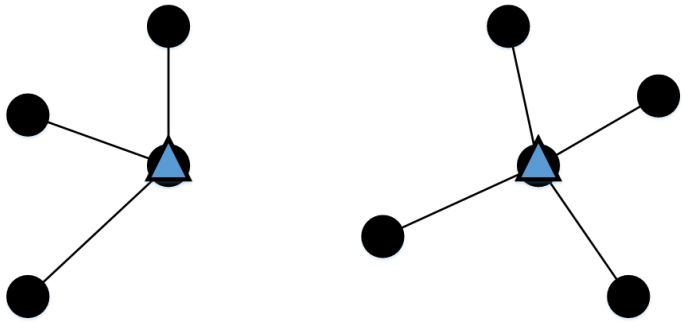
In many of the real life applications listed above, either the facilities have limited service capacities or imposing capacities on the facilities prevents possible overloads, delays and ultimately increases the quality of the service. The p-center problem with capacity restriction of facilities is called the capacitated p-center problem. In the general setting of the capacitated p-center problem, both demands of nodes and capacities of the facilities can be non-identical. If the capacity is equal to the maximum number of nodes that a single facility can serve and this number is identical for each facility, this problem is referred to as the balanced p-center problem. When the facilities have limited capacities, it is worth to discuss different strategies for the allocation of demand nodes. One may require that the demand of each node has to be satisfied by a single facility.



Underlying Network



Absolute



Vertex Restricted

Figure 1.1: Illustration of absolute and vertex restricted 2-center problems on a sample network

Such a requirement results in the single allocation capacitated p-center problem. Another strategy might be to allow that different fractions of the demand of a node can be satisfied by multiple facilities. This problem is referred as the multiple allocation capacitated p-center problem.

Our primary interest in the p-center problem is from a modeling and algorithmic perspective. We focus on both absolute and vertex restricted versions of the p-center problem. We analyze the problem on both weighted and unweighted networks. In addition to the p-center problem, we examine the generalized capacitated p-center problem with non-identical demand and non-identical capacities. We investigate the problem with both single and multiple allocation strategies.

We propose a new formulation and a new method based on this formulation for solving the absolute and vertex restricted p-center problems. We obtain a new integer programming model with better linear programming (LP) bounds by tightening one set of constraints in our model. A semi relaxation of our proposed model gives the tightest lower bound obtained earlier by [1]. We give a polynomial time algorithm to compute the lower bound by solving a finite number of linear programming problems of polynomial size. Additionally, we provide new lower and upper bounds with constant approximation factors and utilize these bounds effectively in our solution methods. The method we propose for solving the p-center problem uses restrictions of the proposed formulation to converge to an optimal solution. While the restriction approach is general enough to allow many variations as dependent on how one chooses restrictions during the process, we focus on a particularly simple restriction which we refer to as the double bound method. One can interpret the double bound method as a generalization of the binary search algorithm. By using the double bound method, we are able to solve some large problems that are reported unsolved in the previous literature. We provide additional larger sized test problems that have not been attempted previously. In addition to the double bound method, we develop a Benders Decomposition algorithm for solving the p-center problem and conduct an experimental study for assessing the performance of this algorithm.

We start our studies on the capacitated p-center problem by developing mathematical formulations. We initially focus on the single allocation case and propose three new mathematical formulations and an exact algorithm that we call as 'successive p-center-allocation' algorithm. We conduct computational experiments on different data sets which contain problems with loose or tight and identical or non-identical capacities. We are able to solve problems with up to 900 nodes while the largest problem solved in the literature has 402 nodes. To our knowledge, there are no studies in the literature that focus on the multiple allocation capacitated p-center problem. We adapt all of our methods that we propose for the single allocation capacitated p-center problem to the multiple allocation case. Additionally, using a non-compact formulation attained through projection, we develop a branch and cut algorithm for the multiple allocation capacitated p-center problem. We are able to solve problems with up to 1291 nodes by using our branch and cut algorithm.

The rest of this thesis is organized as follows. In the next chapter we give the notations and definitions used throughout the thesis. In Chapter 3, we present a detailed review of the related works in the literature. In Chapter 4, we present our mathematical formulations and algorithms for solving the p-center problem. We analyze several relaxations of our mathematical formulations and provide additional lower and upper bounds. Then, we provide a large scale experimental study on our methods for solving the vertex restricted p-center problem on both weighted and unweighted networks. In Chapter 5, we introduce our methods for obtaining new lower and upper bounds for the absolute p-center problem and for generation of intersection points. We present the computational results of the algorithm that we propose in Chapter 4 for solving the absolute p-center problem. In Chapter 6, we introduce our methods for solving the single allocation capacitated p-center problem and provide computational results. In Chapter 7, we give a detailed description of our projected model and the branch and cut algorithm to solve the multiple allocation capacitated p-center problem along with the computational results regarding this problem. We also present the mathematical formulations that we propose for the multiple allocation capacitated p-center

problem. Finally, in Chapter 8, we provide the details of our Benders Decomposition algorithm and conclude with a brief discussion followed by future research directions.

Chapter 2

Notation and Definitions

Let $G = (N, E)$ be a given network with vertex set $N = \{1, \ldots, n\}$ and edge set E. An edge $\{i, j\} \in E$ can be considered as a set of infinite number of points. A point on an edge $\{i, j\} \in E$ is identified by its distance to the endpoints $i, j \in N$. We can think of G as union of its vertices (nodes) and set of all points of all of its edges. Below we present the notation and the definitions that we use in the thesis.

- $l_{ij} > 0$ is the length assigned to each edge $\{i, j\} \in E$.
- $d(x, y) = d_{xy}$ is the length of a shortest path from a point $x \in G$ to another point $y \in G$ in the given network.
- $D(X, i) = \min_{x \in X} d_{xi}$ for any point set $X \subset G$.
- $f(X) = \max_{i \in N} D(X, i)$ for any point set $X \subset G$.

Vertex restricted p-center problem :

- The problem is to find a set $X^* \subseteq N$ of p vertices so that $f(X^*) \leq f(X)$ for any $X \subseteq N$ of p vertices.
- $r_V^* = f(X^*) = \min_{X \subset N : |X| = p} f(X)$ is the optimal value of the vertex restricted p-center problem.

Absolute p-center problem :

- The problem is to find set $X^* \subset G$ of p points so that $f(X^*) \leq f(X)$ for any $X \subset G$ of p points.
- $r_A^* = f(X^*) = \min_{X \subset G : |X| = p} f(X)$ is the optimal value of the absolute p-center problem.
- *Intersection point:* A point x on edge $\{k, m\} \in E$ qualifies as an intersection point if there exist two distinct vertices i and j such that x is the unique point on $\{k, m\}$ for which $d(i, x) = d(x, j)$. Note that $d_{ix} = d_{xj}$ can be achieved with one of the cases $d_{ik} + d_{kx} = d_{xm} + d_{mj}$ or $d_{jk} + d_{kx} = d_{xm} + d_{mi}$ and $d(i, x)$ is the relative radius of x. See Figure 5.1 for an illustration of intersection point.
- P is the set of intersection points in G.

[2] reveals that there exists an optimal solution, say X^*, for the absolute p-center problem such that $X^* \subset (P \cup N)$. There are at most $O(n^2)$ intersection points on any given edge and $O(n^2|E|)$ points on the entire network. Therefore, we can assume that the potential facilities in the absolute p-center come from a finite set.

- $J = \{1, \ldots, m\}$ is the set of potential facilities in G. $J \subseteq N$ for the vertex restricted p-center problem; $J \subseteq (P \cup N)$ for the absolute p-center problem.
- $D = [d_{ij}]$ is the $n \times m$ distance matrix with $i \in N, j \in J$.
- $\rho_1 < \rho_2 < \ldots < \rho_M$ is an ordering of the distinct distance values of D.
- $R = \{\rho_1, \rho_2, \ldots, \rho_M\}$.
- $T = \{1, \ldots, M\}$ is an index set associated with R.
- $N_r(i) = \{j \in J : d_{ij} \leq r\}$ is the set of accessible nodes from node $i \in N$ within radius r.

Capacitated p-center problem:

- h_i : The demand of node $i \in N$.
- K_j : The capacity of node $j \in J$.
- $K_j^r = \min\{K_j, \sum_{j \in N_r(i)} h_j\}$: The *effective capacity* [3] of node $j \in J$ for radius r.

Weighted p-center problem:

- w_i : The positive weight associated with each vertex $i \in N$.
- $f_w(X) = \max_{i \in N} w_i D(X, i)$ for any point set $X \subset G$.
- *Intersection point:* A point $x \in G$ qualifies as an intersection point if there exist two distinct vertices i and j such that $w_i d(i, x) = w_j d(x, j)$ and there exists a positive real number ε such that $\max\{w_i d(i, x'), w_j d(j, x')\} > w_i d(i, x)$ for all points x' for which $0 < d(x, x') < \varepsilon$.

Finally, we define the set covering problem, which is closely related to the uncapacitated p-center problem. Given a zero-one matrix $A = [a_{ij}]$, the set covering problem is to find a set of columns at minimal cost that cover the rows of the A. In order to minimize the number of facilities required to serve all demand nodes within a given radius value r, one can solve a set covering problem SC(r) by constructing A as follows:

$$a_{ij} = \begin{cases} 1, & \text{if } d_{ij} \leq r, \quad (w_i d_{ij} \leq r \text{ for the weighted case}) \\ 0, & \text{otherwise} \end{cases} \qquad \forall i \in N, j \in J$$

for all $i \in N, j \in J$. If the optimal value of SC(r) is greater than p, then this means that the optimal value of the p-center problem is greater than r; if it is less than or equal to p, then this means that the optimal value of the p-center problem is less than or equal to r.

Chapter 3

Literature Review

In this chapter, we list the related works in the literature for the p-center problem and the capacitated p-center problem. Section 3.1 consists of the absolute and vertex restricted p-center problem studies and Section 3.2 is reserved for the capacitated p-center problem studies. In Section 3.3, we provide a brief summary of our contribution to the literature with this thesis work.

3.1 Absolute and Vertex Restricted p-Center Problems

The absolute 1-center problem is initially defined by Hakimi [4] and solved using graphical methods by taking advantage of the piecewise linearity of the function $f(X)$ on any edge $\{k, m\} \in E$, that is, X is a subset of the points on edge $\{k, m\} \in E$. Piecewise linearity for the absolute 1-center problem has important consequences for $p > 1$ as it leads to the existence of a finite point set $P \subset G$ such that there exists an absolute p-center in $(P \cup N)$. This is initially observed by Minieka [2] and extended to the weighted case by Kariv and Hakimi [5]. This property is generalized later by Hooker et al. [6] to a more general setting.

Existing solution methods are either based on solving a sequence of set covering problems or enumerating p-element subsets of J. The first set-covering based approach is proposed by Minieka [2] for the absolute p-center problem. Minieka [2] presents a systematic method to update the set covering matrix to converge to an optimal solution in a finite number of steps. At each step the set covering problem is solved for the updated matrix corresponding to a smaller distance value selected from the distinct distance values set R and the algorithm is terminated when the optimal value of the set covering problem is greater than p. Christofides and Viola [7] solve the weighted absolute p-center problem by first generating regions in the network. A region is a set of points (a single point or an edge segment) that can reach the same set of vertices within radius r. Then, a bipartite graph is constructed in the following form. The original nodes are put on one side, the regions are put on the other side, and there is an edge between a node and a region if the node is reachable by that region. Finally, they find a minimal covering of the constructed bipartite graph. This approach does not make use of the finite distance set R, instead it proposes increasing the radius value by a small increment at each iteration. Toregas et al. [8] solve the vertex restricted p-center problem by solving a linear programming relaxation of the associated set covering problem and adding a cut in case of fractional solutions. Garfinkel et al. [9] solve the absolute p-center problem by solving a sequence of set covering problems but they first reduce the search space by finding a heuristic solution X and eliminating from J all those points whose relative radii are greater than $f(X)$. They apply two types of tests to eliminate some of the intersection points and standard matrix reductions and heuristic techniques to reduce the number of rows and columns of the set covering matrix. Their method can be extended to the weighted p-center problem as well. Sac [10] implements a new set covering based algorithm for solving the absolute p-center problem. In this algorithm, both the construction of the set covering problems used and the generation of the intersection points differ from the traditional methods in the literature. The new algorithm is compared with the classical set covering based binary search algorithm on problems with up to 900 nodes and better solution times are observed with the new method.

Daskin [11] gives the first IP formulation of the vertex restricted p-center problem but prefers to use a set covering based bisection search over an interval defined by pre-computed lower and upper bounds on r_V^*. Daskin [12] improves this algorithm by solving, via Lagrangean relaxation, a maximal covering location problem in which the total number of vertices that are covered within r is maximized while the number of open facilities is restricted to p. Ilhan and Pinar [13] propose a two phase extension of Daskin's [11] algorithm for the vertex restricted problem. In the first phase, several LP relaxations of the set covering problem are solved as feasibility problems by forcing the objective value to be less than or equal to p to find an appropriate lower bound on r_V^*. In the second phase, several set covering problems with the same restriction on the objective value are solved by systematically changing the radius value starting from this lower bound. Al-khedhairi and Salhi [14] propose some modifications to the algorithms in [11] and [13]. Elloumi et al. [1] propose a different IP formulation for the p-center problem. They give a lower bound which is tighter than the LP relaxations of both models in the literature and a polynomial time method to compute it via solving a sequence of LPs. They solve the p-center problem by performing a binary search over the ordered list of distinct values of distances that are between their proposed lower and upper bounds. A set covering problem is solved for each selected distance value between the bounds. They solve problems from the OR-Library [15] and TSPLIB [16] with up to 1817 nodes using binary search. To our knowledge, this is the largest network size solved in the literature.

We note here that the term "bisection search" refers to successively halving a real interval and discarding either the lower or the upper half in each step until its size is smaller than a predetermined positive real number whereas the term "binary search" refers to performing essentially the same operation on a finite list of numbers using a median element of the list.

Kariv and Hakimi [5] prove that the weighted p-center problem is NP-Hard even if the network G is planar with unit vertex weights, unit edge lengths and with the maximum vertex degree of 3. They provide enumeration based algorithms for the weighted and unweighted p-center problem for $p = 1$ and $p > 1$ on general and tree networks. The computational complexities of these algorithms

are $O(|E|n \log n)$ for the weighted absolute 1-center, $O(|E|n + n^2 \log n)$ for the unweighted absolute 1-center, $O(|E|^p(n^{2p-1})/(p-1)! \log n)$ for the weighted absolute p-center, and $O(|E|^p(n^{2p-1})/(p-1)!)$ for the unweighted absolute p-center problems on general networks. Moreno [17] provides another enumeration based algorithm for the weighted p-center problem with better computational complexity of $O(|E|^p n^{p+1} \log n)$. Later, Tamir [18] provides improved complexity bounds for the weighted and unweighted p-center problems by combining the algorithms of [5] and [17]. The improved bounds are $O(n^p|E|^p \log^2 n)$ and $O(n^{p-1}|E|^p \log^3 n)$ for the weighted and unweighted problems, respectively.

There are several studies that focus on solving special cases of the p-center problem such as 1-center problem and p-center problem on tree networks. Goldman [19] comes up with a localization theorem for the absolute 1-center problem. This theorem results in a very efficient polynomial algorithm for the tree networks and after this pioneering study, tree networks have received considerable attention. Handler [20] proves that the absolute center of a tree is the midpoint of a longest path in the tree when all nodes have unit weight and provides a polynomial algorithm for finding the absolute center of a tree. Halfin [21] proposes a modification for the algorithm of [19]. Dearing and Francis [22] study weighted absolute 1-center problem on both tree and general networks. Hakimi et al. [23] study weighted cases of absolute 1-center and absolute p-center problems on tree networks. Low order polynomial time algorithms for solving the p-center problem on tree networks are provided by [5], [24], [25], [26], [27], [28]. Another study on the absolute 1-center problem by [29] proposes an improvement on the algorithm of [4]. The improvement is proposed on finding the best candidate point on an edge. The proposed method finds the best candidate point by using only the shortest path distance values between node pairs and does not require the knowledge of point-vertex distance function. Recently, Dvir and Handler [30] propose a new algorithm for solving absolute 1-center problem on general networks.

Handler and Mirchandani [31] come up with a relaxation method for solving the p-center problem. Chen and Chen [32] propose new algorithms based on the relaxation method provided by [31] for solving the p-center problem. Caruso et

Table 3.1: Exact solution methodologies for the p-center problem on general networks

	Author(s)	Notes
$p = 1$	Hakimi [4]	Vertex and absolute 1-center, graphical method
	Dearing and Francis [22]	Weighted absolute 1-center
	Hakimi et al. [23]	Weighted absolute 1-center
	Kariv and Hakimi [5]	Complexity result, enumeration algorithm
	Minieka [29]	Improvement for [4]
	Dvir and Handler [30]	Improvement algorithm
$p > 1$	Minieka [2]	Finite set of potential facilities
	Christofides and Viola [7]	Regions and reachable nodes
	Toregas et al. [8]	Set covering based LP relaxations
	Garfinkel et al. [9]	Improvement for [2]
	Kariv and Hakimi [5]	Complexity result, enumeration algorithm
	Handler and Mirchandani [31]	Relaxation method
	Moreno [17]	Improved enumeration algorithm
	Tamir [18]	Combination of [5] and [17]
	Daskin [11]	First MIP model, set covering based bisection algorithm
	Daskin [12]	Improvement for [11]
	Ilhan and Pinar [13]	Improvement for [11]
	Caruso et al. [33]	Vertex restricted p-center problem
	Elloumi et al. [1]	IP model, set covering based binary search algorithm
	Al-khedhairi and Salhi [14]	Improvement for [11] and [13]
	Chen and Chen [32]	Relaxation based algorithm
	Sac [10]	Improvement algorithm

al. [33] propose an algorithm, named Dominant, for the unweighted vertex p-center problem. They provide four versions of this algorithm (QuickDominant (QD), RandomDominant (RD), ExactDominant (ED), BestDominant (BD)) and two of them (ED and BD) can solve the problem optimally. They compare the performance of their algorithms with a software package developed by Daskin [11]. This package can solve the p-center problems up to 150 nodes to optimality. The comparison shows that ED and BD outperform the package by Daskin [11] in terms of computational time in all cases.

Tables 3.1 and 3.2 summarizes the exact solution methodologies for the p-center problem on general networks and tree networks, respectively.

Other than the exact solution methodologies presented above there are a certain number of heuristic algorithms proposed for solving the p-center problem.

Table 3.2: Exact solution methodologies for the p-center problem on tree networks

	Author(s)	Notes
$p = 1$	Dearing and Francis [22]	Weighted absolute 1-center
	Goldman [19]	Localization theorem for absolute 1-center
	Handler [20]	Absolute center of a tree
	Halfin [21]	Modification of [19]
	Hakimi et al. [23]	Weighted absolute 1-center
$p > 1$	Hakimi et al. [23]	Weighted absolute p-center
	Kariv and Hakimi [5]	Polynomial time algorithm
	Megiddo et al.[24]	Polynomial time algorithm
	Tansel et al. [25]	Polynomial time algorithm
	Megiddo and Tamir [26]	Polynomial time algorithm
	Jaeger and Kariv [27]	Polynomial time algorithm
	Shaw [28]	Polynomial time algorithm

Gonzalez [34] proposes a 2-approximation algorithm for the vertex restricted p-center problem. The computational complexity of this algorithm is $O(pn)$ and it works under the triangular inequality assumption. Another 2-approximation algorithm with $O(|E|log|E|)$ complexity for the unweighted vertex p-center problem is constructed by Hochbaum and Shmoys [35]. The algorithm works under the triangular inequality and complete graph assumptions. Since [36] and [37] reveal that p-center problem with triangle inequality is NP-complete and any δ-approximation for $\delta < 2$ is NP-hard, this algorithm is a best possible heuristic, that is, it produces solutions within the best approximation factor. Plesnik [38] generalizes the study in [35] and introduces a polynomial time algorithm for the p-center problem on vertex-weighted networks. This algorithm achieves a worst-case error ratio of 2. By slightly modifying this algorithm, another algorithm with the worst-case error ratio of 2 for the absolute p-center problem is obtained. Shmoys [39] introduces a relaxed version of the algorithm proposed in [35]. This algorithm considers the following decision problem: does there exist a set of p vertices that will cover all demand points within the radius value of r. It either identifies that there is no feasible solution for the specified radius value, r, or if there exist one, it generates a solution of p centers within the radius value of $2r$. The algorithm produces solutions whose objective values are at most two times the optimal by employing a bisection search on the radius values.

The first metaheuristic approach for the p-center problem is due to Mladenović et al. [40]. They develop a vertex substitution local search, a chain substitution Tabu Search (TS), and a variable neigborhood search (VNS) for solving unweighted p-center problem without the triangular inequality assumption. The computational experiments show that when compared with the heuristic algorithm of [35], all heuristics proposed in [40] perform better. Salhi and Al-Khedhairi [41] provide a tree-level metaheuristic, which is a combination of variable neighborhood search and perturbation schemes and produces tight upper and lower bounds, to solve the unweighted vertex restricted p-center problem. By utilizing these bounds on Daskin's [11] algorithm, they improve the computational time of the algorithm. The heuristic starts with generation of initial feasible solutions. The authors employ two types of neighborhood structures and use them consecutively until no better solution can be obtained (they allow movements to infeasible solutions). Then a perturbation mechanism is introduced and if a new solution is obtained, the algorithm repeats the earlier steps of visiting the neighborhoods. Pullan [42] puts forward a population based heuristic (PBS) for the vertex p-center problem. This algorithm incorporates a population based metaheuristic (a genetic algorithm) with an iterative improvement local search methodology. The genetic algorithm produces several qualified starting points for the improvement heuristic and the local search explores the neighborhoods of these initial solutions effectively. The computational study on the benchmark instances show that this algorithm performs quite satisfactorily in terms of robustness, quality and computational effort required. Its comparison with [40]'s algorithm show that PBS performs much better. Recently, Davidović et al. [43] apply a bee colony optimization method to the vertex restricted p-center problem.

A vertex closing approach for the p-center problem on complete networks with distance values that satisfy the triangular inequality is put forward by Martinich [44]. Initially all vertices in the network are considered as centers and they need to be closed until p of them remain. The main idea behind the strategy to close the vertices is to select the ones that are close to the open centers. The optimal set of vertices to close is characterized by the embedded sub-graphs of the original graph. The author analyzes the properties of these sub-graphs and

obtains initial lower and upper bounds. As a byproduct of this analysis, two polynomial time algorithms are constructed. For some special cases it is proven that these algorithms converge to optimal solution. The computational studies indicate satisfactory performance of the algorithms. When compared with the 2-approximation algorithm in [35], the proposed algorithms outperform in terms of the number of instances solved optimally even though they do not guarantee a 2-optimal solution. Bozkaya and Tansel [45] show that there exists a spanning tree of any connected network such that the optimal p-center of this tree is optimal also for the network under consideration. They conduct experiments on two classes of spanning trees to observe how often these trees provide the optimal solution. They conclude that these two classes of spanning trees do not always include the optimizing tree, but they do in most of the problems. Mihelič and Robič [46] focus on the vertex restricted p-center problem and propose a heuristic algorithm based on solving a sequence of dominating set problems. The experimental comparison with a pure greedy heuristic of $O(n^2p)$ and the heuristic algorithms of [34], [39], and [35] reveals that their algorithm performs better than the previously implemented ones. A polynomial time heuristic algorithm for the minimum dominating set problem, which is commonly utilized in solving the p-center problems is introduced by Robič and Mihelič [47]. By applying this algorithm, they obtain a polynomial time heuristic for solving the vertex p-center problem (complete networks with triangular inequality). The computational experiments performed on 40 standard test problems indicate that their algorithm performs much better than the other heuristics in the literature and competes with the best known algorithms.

Table 3.3 provides a brief summary of the heuristic methods for the p-center problem.

A review of the early theory and algorithms for network location problems, including the p-center problem, is given in [48] and [49] (see also [31], [11], [50], and [51]). Various theoretical and algorithmic aspects of the p-center problem for general and tree networks are discussed in [52].

Table 3.3: Heuristic methodologies for the p-center problem

	Author(s)	Notes
Approximation algorithms	Gonzalez [34]	2-app. for vertex p-center
	Hochbaum and Shmoys [35]	2-app. for vertex p-center
	Plesnik [38]	2-app. for weighted vertex and absolute p-center problems
	Shmoys [39]	Improvement for [35]
Meta-heuristic algorithms	Mladenović et al. [40]	Local Search, TS, VNS
	Salhi and Al-Khedhairi [41]	Three-level meta-heuristic
	Pullan [42]	Incorporates a genetic algorithm and local search
	Davidović et al. [43]	Bee colony optimization
Other methods	Martinich [44]	Vertex closing approach
	Bozkaya and Tansel [45]	Spanning trees
	Mihelič and Robič [46]	Dominating set problems
	Caruso et al. [33]	Set covering problems
	Robič and Mihelič [47]	Minimum dominating set problem

3.2 Capacitated p-Center Problem

The first study on the capacitated p-center problem is by Bar-Ilan et al. [53]. They study the the p-center problem where a facility can serve at most L demand nodes. They refer to this problem as the balanced p-center problem and this is a special case of the problem that we study where each node has a demand of one unit and each facility has a capacity of L units. They provide an approximation algorithm with an approximation factor of 10. Khuller and Sussmann [54] study the same problem and provide an approximation algorithm with an approximation factor of 6. They also study a simplified version of this problem, in which more than one center can be placed on a node. They call this problem as the capacitated multi-p-center problem and propose a 5-approximation algorithm for it. Cygan et al. [55] focus on the same problem but with non-identical capacities, that is, the capacities of the facilities are not identical. They prove that there exits a constant factor approximation algorithm for this problem; they do not give the exact constant, but state that it is in the order of hundreds. They also provide an 11-approximation algorithm for the capacitated p-center problem with non-uniform capacities when more than one center can be placed on a node.

Their algorithm uses an LP rounding technique. Jaeger and Goldberg [56] provide a polynomial time algorithm for the capacitated p-center problem on trees with identical capacities. Scaparra et al. [57] present a large-scale neighborhood search heuristic for the capacitated p-center problem. Although they also present a mixed integer programming (MIP) formulation for the problem, they focus on their heuristic algorithm and solve problems with up to 402 nodes by using this algorithm.

In addition to the mathematical model in [57] there are only two studies that attempt to solve the capacitated p-center problem optimally. The first one is due to Özsoy and Pınar [58]. They provide an exact algorithm, which is a modification and adaption to the capacitated case of the 2-Phase algorithm in [13] proposed for the uncapacitated p-center problem. In this algorithm, the capacitated concentrator location problem and the bin-packing problem are used as sub-problems. In the first phase of the algorithm, the LP relaxation of the capacitated concentrator location problem is solved for different radius values iteratively and a lower bound for the optimal value of the capacitated p-center problem is obtained. In the second phase of the algorithm either the capacitated concentrator location problem or the bin-packing problem is solved initially for the smallest distance value which is greater than or equal to the lower bound. The objective value of the sub-problem gives the number of facilities to be opened to satisfy capacities for the given radius value. Therefore, if the optimal value obtained from the sub-problem is greater than p, the sub-problem is solved for the next smallest distance value which is larger than the current distance value. This procedure is repeated until the smallest distance value that provides an optimal value that is less than or equal to p is achieved. They solve problems with up to 402 nodes optimally. They also present a MIP formulation for the capacitated p-center problem, but they do not provide any computational study on this model.

The second exact algorithm is provided by Albareda-Sambola et al. [59]. They obtain lower bounds from the Lagrangean duals based on two auxiliary problems: The maximum demand coverage within fixed radius problem and the minimum required centers within fixed radius problem. Their exact algorithm solves the

Table 3.4: Related works on the capacitated p-center problem

Author(s)	Notes
Bar-Ilan et al. [53]	10-app. for balanced p-center
Jaeger and Goldberg [56]	Identical capacities on tree networks, polynomial time algorithm
Khuller and Sussmann [54]	6-app. for balanced p-center, 5-app for balanced p-center with multi centers
Scaparra et al. [57]	MIP model, neighborhood search heuristic
Özsoy and Pınar [58]	MIP model, modification and adaptation of [13]
Albareda-Sambola et al. [59]	Lagrangean duals based bounds, Binary search algorithm based on an auxiliary problem
Cygan et al. [55]	Finite app. factor for non-identical capacities, 11-app. for the problem with multi centers

second auxiliary problem and selects the radius value to solve this problem from the set of possible radius values by using a binary search strategy. The set of radius values is restricted by the lower and upper bounds they obtain. They solve problems with up to 402 nodes optimally.

Table 3.4 provides a list of the related works on the capacitated p-center problem in the literature.

3.3 Contribution of the Thesis Work

In this thesis, we initially focus on the vertex restricted p-center problem. We provide a new mathematical formulation and a new method based on successive restrictions of the new formulation. We obtain a new integer programming model with better linear programming (LP) bounds by tightening one set of constraints in our model. A semi relaxation of our proposed model gives the tightest known lower bound, which is obtained earlier by Elloumi et al. [1], and we present a polynomial time algorithm to compute this bound. Additionally, we propose new lower and upper bounds, which are within a constant multiple of the optimal value of the problem and can be obtained via polynomial time algorithms. We conduct experiments on weighted and unweighted benchmark problems from OR-Library [15] and TSPLIB [16] with up to 3038 nodes by using a specialization of

our method, referred to as double bound algorithm. We solve the problems that require large amount of time by integrating the reduction rules to our algorithm and observe significant improvements in utilization of the reduction rules. As our methods are applicable to both vertex restricted and absolute p-center problems, we focus on solving the absolute p-center problem by using the double bound method. We devise new theoretical results for the absolute p-center problem and use these results to develop a new method for generating the intersection points. We solve problems from OR-Library [15] with up to 900 nodes and 16056 edges. In addition to the double bound method, we develop another exact algorithm based on the Benders Decomposition method to solve the p-center problem. We compare the performances of the Benders algorithm and the double bound algorithm on problems from OR-Library [15]. In addition to the uncapacitated p-center problem, we focus on the capacitated p-center problem. For the single allocation capacitated p-center problem, we propose new mathematical formulations and a new exact algorithm that solves the uncapacitated p-center problem and an allocation problem successively. We refer to this algorithm as "successive p-center-allocation algorithm" and this algorithm differs from the approaches in the literature since we decompose the problem into two different and relatively easier problems and solve them iteratively to obtain the optimal solution for the capacitated p-center problem. Moreover, this thesis focuses on the multiple allocation capacitated p-center problem for the first time in the literature. The formulations and the successive p-center-allocation algorithm that we propose for the single allocation capacitated p-center problem are readily applicable to the multiple allocation capacitated p-center problem. Additionally, we propose a branch and cut algorithm based on a non-compact formulation obtained through projection for solving the multiple allocation capacitated p-center problem. We conduct large scale experiments by using our algorithms and solve problems with up to 900 nodes and 1291 nodes for single and multiple allocation capacitated p-center problems, respectively. The dimensions of problems we solve in this thesis are significantly higher than the ones reported in the literature.

Chapter 4

The Vertex Restricted p-Center Problem

In this chapter, we first present the existing mathematical formulations in Section 4.1. Then, we propose a new mathematical formulation and a tightened version of our formulation in Section 4.2. In Section 4.3, we make a comparison between the LP relaxations of our formulations and the previous formulations. We present a lower bound that we obtain from our IP formulation by relaxing the binary restriction on one set of variables. We prove that this bound is equivalent to the tightest known lower bound in the literature and provide a polynomial time algorithm to obtain this bound. In addition to the relaxation bounds, we provide new lower and upper bounds which can be obtained very quickly. In Section 4.4, we first give the underlying idea of our method, and then we give the general structure of our double bound algorithm as the solution methodology. We introduce six variations of the double bound algorithm. In Section 4.5, we provide the experimental results obtained from the mathematical formulations and our algorithms. We solve problems from OR-Library [15] and TSPLIB [16] in these experiments. Finally, we provide concluding remarks in Section 4.6. A version of this chapter has appeared in [60].

4.1 Mathematical Formulations Existing in the Literature

The first model for the p-center problem in the literature is proposed by Daskin [11]. Define a binary variable y_j with $y_j = 1$ if a center is placed at vertex $j \in J$ and 0 otherwise. Define binary variables x_{ij} to be 1 if $i \in N$ assigns to a center placed at $j \in J$ and 0 otherwise. This formulation referred to as P1 in the sequel, is as follows:

$$
\begin{aligned}
(\text{P1}): \quad & \min \; z && && (4.1)\\
& \text{s.t.} \; \sum_{j \in N} d_{ij} x_{ij} \leq z && \forall i \in N, && (4.2)\\
& \sum_{j \in J} x_{ij} = 1 && \forall i \in N, && (4.3)\\
& x_{ij} \leq y_j && \forall i \in N, j \in J, && (4.4)\\
& \sum_{j \in J} y_j \leq p, && && (4.5)\\
& y_j \in \{0,1\} && \forall j \in J, && (4.6)\\
& x_{ij} \in \{0,1\}, && \forall i \in N, j \in J. && (4.7)
\end{aligned}
$$

Constraints (4.3) assign each vertex to exactly one center and (4.1) and (4.2) ensure that the objective value is no less than the maximum vertex-to-center distance. Constraints (4.4) ensure that no vertex assigns to j unless there is a center at j. Constraint (4.5) restricts the number of centers to p. Constraints (4.6) and (4.7) are the binary restrictions.

The second IP formulation is due to Elloumi et al. [1]. Their formulation is similar to a canonical representation of the simple plant location problems given earlier by [61]. Define y_j to be the same as in P1 and the additional binary variables u^k, $k = 2, \ldots, M$, with $u^k = 0$ only if all vertices can be covered within a radius value of ρ_{k-1} and $u^k = 1$ otherwise. Denote by P2 the formulation of [1]

given below:

$$\text{(P2)}: \quad \min \ \rho_1 + \sum_{k=2}^{M} (\rho_k - \rho_{k-1}) u^k \tag{4.8}$$

$$\text{s.t.} \ \sum_{j \in J} y_j \geq 1 \tag{4.9}$$

$$u^k + \sum_{j: d_{ij} < \rho_k} y_j \geq 1 \qquad \forall i \in N, k = 2, \ldots, M \tag{4.10}$$

$$u^k \in \{0, 1\} \qquad k = 2, \ldots, M. \tag{4.11}$$

$$(4.5), (4.6)$$

Constraint (4.9) is required to eliminate null solutions (with no center). Constraints (4.10) and the objective function (4.8) ensure that all vertices are covered by their closest centers.

4.2 Proposed Formulations

We now propose a new formulation of the p-center problem. Exactly one of the radius values in R determines the optimal value of the p-center problem. Associate a binary variable z_k with ρ_k, $k \in T \equiv \{1, \ldots, M\}$ with $z_k = 1$ if ρ_k is selected as the optimal value and 0 if not. For $i \in N = \{1, \ldots, n\}, j \in J = \{1, \ldots, m\}$ and $k \in T$, define $N_{\rho_k}(i) = \{j \in J : d_{ij} \leq \rho_k\}$. We use the variables y_j as before; that is, $y_j = 1$ if a center is placed at site j and 0 otherwise. The proposed formulation, referred to as P3, is as follows:

$$\text{(P3)}: \quad \min \ \sum_{k \in T} \rho_k z_k \tag{4.12}$$

$$\text{s.t.} \ \sum_{j \in N_{\rho_k}(i)} y_j \geq z_k \qquad \forall i \in N, \forall k \in T \tag{4.13}$$

$$\sum_{k \in T} z_k = 1 \tag{4.14}$$

$$z_k \in \{0, 1\} \qquad \forall k \in T. \tag{4.15}$$

$$(4.5), (4.6)$$

Constraint (4.14) ensures that exactly one of the variables z_k is selected as 1 and the objective function (4.12) determines the optimal value as the corresponding value ρ_k. Constraints (4.13) ensure that each vertex is covered within the selected radius by at least one center. Constraints (4.15) are binary restrictions.

For any feasible solution (y, z) of P3, we can obtain a feasible solution (y, u) for P2, which provides exactly the same objective value, by setting

$$u^k = \sum_{q=k}^{M} z_q, \qquad k = 2, \ldots, M. \tag{4.16}$$

The reverse can also be achieved by using

$$\begin{aligned} z_k &= u^k - u^{k+1}, \qquad k = 2, \ldots, M-1, \\ z_M &= u^M, \\ z_1 &= 1 - u^2. \end{aligned} \tag{4.17}$$

By using this relationship, we can obtain a tighter constraint for (4.13). When we consider all distinct distance values in the increasing order, (4.10) implies $u^k + \sum_{j:d_{ij} \le \rho_{k-1}} y_j \ge 1, \forall i \in N, k = 2, \ldots, M$. Replacing u^k with $\sum_{q=k}^{M} z_q$ and the right hand side with $\sum_{q=1}^{M} z_q$ we obtain $\sum_{j:d_{ij} \le \rho_k} y_j \ge \sum_{q=1}^{k} z_q, \forall i \in N, k = 1, \ldots, M-1$. For $k = M$, $\sum_{j:d_{ij} \le \rho_k} y_j = \sum_{j \in J} y_j \ge 1 = \sum_{q=1}^{k} z_q$. Then we can replace (4.13) with

$$\sum_{j \in N_{\rho_k}(i)} y_j \ge \sum_{q=1}^{k} z_q, \forall i \in N, \forall k \in T. \tag{4.18}$$

The new formulation, referred to as (P4), with the tightened constraints is basically as follows:

$$\begin{aligned} \text{(P4)}: \qquad & \min (4.12) \\ & \text{s.t. } (4.5), (4.6), (4.14), (4.15), \text{ and } (4.18). \end{aligned}$$

P3 and P4 has $m+M$ binary variables and $nM+2$ constraints. P2 has $m+M-1$ binary variables and $n(M-1)+2$ constraints. On the other hand, P1 has one real variable, $m+mn$ binary variables and $2n+mn+1$ constraints. Since M is at

most mn, P2, P3 and P4 have $O(mn)$ binary variables and $O(mn^2)$ constraints, which is $O(n)$ times more than the number of constraints of P1. We compare the computational performance of our formulations with P1 and P2 in Section 4.5.

4.3 Relaxation and Heuristic Bounds

Before solving P3, if we know that there exists a set $S \subset R$ such that the optimal value of the p-center problem is different from any $\rho_j \in S$, we can effectively use this information: we remove these values from R and drop associated z_j variables from the model, thus, decrease the size of the problem to be solved. For example, if we have a lower bound LB on the optimal objective value, then we may remove any $\rho_j < LB$ from R; similarly, if we have an upper bound UB, then we may remove any $\rho_j > UB$ and solve the model with the restricted R and obtain an optimal solution. In this section, we first analyze the LP relaxation bounds of the four models discussed in this chapter. Then, we propose a tighter bound obtained from a partial relaxation of our formulation P3. In addition to the lower bounds that we obtain from relaxations, we propose new lower and upper bounds that can be obtained efficiently.

4.3.1 LP Relaxations

Let LP1, LP2, LP3 and LP4 denote the LP relaxations of P1, P2, P3 and P4, respectively and $val(LP1), val(LP2), val(LP3)$ and $val(LP4)$ denote their optimal values. Elloumi et al.[1] showed that the LP bound of P2 is as good as the LP bound of P1. From (4.16) and (4.17) we know that there is a one-to-one correspondence between the feasible solutions of P2 and P4. Obviously, this is valid for also the LP relaxations of P2 and P4, that is, for any feasible solution (y, u) of P2, there is a corresponding feasible solution (y, z) of P4 with the same objective value and vice versa. Therefore, the LP bounds of P2 and P4 are equivalent and they are as good as the LP bound of P1. Since any feasible solution to the LP4 is also feasible for the LP3, $val(LP3) \leq val(LP4)$. However, the reverse might not

be true and we are able to find problems that support otherwise. On the other hand, the LP bounds of P1 and P3 are not comparable.

4.3.2 Semi Relaxations

Lower bounds based on LP relaxations of the set covering problem are generated for various values of r and used in a bisection search by the algorithm of Ilhan and Pinar [13]. Elloumi et al. [1] propose a lower bound LB* which is tighter than the LP relaxation bounds of P1 and P2. LB* is obtained from P2 by relaxing the integrality restrictions on the variables $y_j, j \in J$, while retaining all other constraints of P2. LB* requires solving a mixed integer program, but Elloumi et al. [1] additionally give a method to compute LB* that requires solving a polynomial number of linear programming problems of polynomial size.

We now propose a relaxation bound based on P3 and prove that this bound is equal to the tightest known bound (the bound LB* in [1]). Let RP2, RP3 and RP4 be the relaxations that retain the objective function and all constraints of P2, P3 and P4, respectively, except that the constraints $y_j \in \{0,1\}, j \in J$, are replaced with the constraints $y_j \geq 0, j \in J$. LB* is the optimal value of RP2. Let $val(RP3)$ and $val(RP4)$ be the optimal values of RP3 and RP4, respectively. The equivalence of LB* and $val(RP4)$ is obvious. Moreover, one can directly see that $val(RP3) \leq val(RP4)$ since any (y, z) that satisfies (4.18) satisfies (4.13) as well. To prove that $val(RP4) \leq val(RP3)$, let us consider an optimal solution $(\overline{y}, \overline{z})$ for RP3 with $val(RP3) = \rho_{k'}$. Then $\overline{z}_{k'} = 1$ and $\overline{z}_k = 0$ for $k \neq k'$. In this case, $\sum_{q=1}^{k} \overline{z}_q = 0$ for $k < k'$ and $\sum_{q=1}^{k} \overline{z}_q = 1$ for $k \geq k'$. Since $\sum_{j \in N_{\rho_k}(i)} \overline{y}_j \geq \sum_{j \in N_{\rho'_k}(i)} \overline{y}_j \geq 1, \forall i \in N, k > k'$, $(\overline{y}, \overline{z})$ is feasible for RP4 and this implies that $val(RP4) \leq val(RP3)$. Thus, we can conclude that $val(RP3) = val(RP4) =$ LB*.

A direct computation of the proposed lower bound requires solving a mixed integer linear program, RP3, with M binary variables. We now give an alternative method which works in polynomial time. For fixed $h \in T$, add the single

constraint $z_h = 1$ to problems P3 and RP3 and call the resulting problems P_h and RP_h, respectively. Let $val(P_h)$ and $val(RP_h)$ be the optimal values of P_h and RP_h, respectively. In case of infeasibility, $val(.)$ is taken to be ∞.

Proposition 1. $val(RP3) = \min_{h \in T} val(RP_h)$.

Proof. We have $val(RP3) \leq val(RP_h), \forall h \in T$ since RP_h is a restriction of RP3. Now, we show that equality is achieved by some $h \in T$. Let $val(RP3)$ be ρ_a for some $a \in T$. Then there is an optimal solution (y', z') to RP3 such that $z'_k = 1$ for $k = a$ and $z'_k = 0$ for $k \in T \setminus \{a\}$. This implies that (y', z') is a feasible solution to RP_a and its objective value is ρ_a. Hence, $val(RP_a) \leq \rho_a$ due to the feasibility of (y', z') to RP_a. We also have $\rho_a = val(RP3) \leq val(RP_a)$ from the first line in the proof. Hence, $val(RP3) = val(RP_a) = \min_{h \in T} val(RP_h)$. □

A closer examination of RP_h shows that it is a linear program in recognition form. To justify this, consider first the problem P_h. Since P_h has all constraints of P3 plus the new constraint $z_h = 1$, constraint (4.14) and $z_h = 1$ imply that $z_k = 0, \forall k \in T \setminus \{h\}$ in every feasible solution. With $z_h = 1$ and $z_k = 0$ for $k \in T \setminus \{h\}$, constraints (4.14) and (4.15) become redundant and can be dropped. Substituting the values of the z-variables in (4.12) results in a constant objective value of ρ_h. Substituting $z_k = 0$ for $k \in T \setminus \{h\}$ in constraints (4.13) makes all constraints in (4.13) redundant except those corresponding to $i \in N$ and $k = h$. It follows that P_h is the following integer program in recognition form: Find $y \in \{0, 1\}^m$, such that

$$\sum_{j \in N_{\rho_h}(i)} y_j \geq 1 \qquad \forall i \in N, \tag{4.19}$$

$$\sum_{j \in J} y_j \leq p. \tag{4.20}$$

RP_h is obtained from P_h by relaxing the binary restriction on y and replacing it with $y_j \geq 0, j \in J$. Hence, RP_h is the following LP in recognition form: Find $y \in \mathbb{R}^m$, if it exists, such that $y \geq 0$ and y satisfies (4.19) and (4.20).

To compute $val(RP3)$, it suffices to compute $\min_{h \in T} val(RP_h)$. This is achieved

in polynomial time by solving $O(log_2 M)$ linear programs RP_h for each ρ_h selected from R during a binary search (Algorithm 1).

Algorithm 1 BINARY

$\rho_1 < \rho_2 < \ldots < \rho_M$, min $\leftarrow 1$, max $\leftarrow M$, and $LB \leftarrow \infty$.
while max $-$ min ≥ 1 **do**
  $mid \leftarrow \lfloor (\min + \max)/2 \rfloor$,
  Solve RP_{mid}.
  if RP_{mid} is feasible **then**
    $LB \leftarrow \rho_{mid}$,
    max $\leftarrow mid$.
  else
    min $\leftarrow mid + 1$.
  end if
end while

We solve RP3, equivalently RP4, on 40 p-median instances from the OR-Library [15] by using the algorithm BINARY and present results in Table 4.1. In these computations, we restrict R to $R \cap [LB2, UB2]$ where LB2 and UB2 are the bounds that we shall explain in Section 4.3.3. We observe that in 36 of these instances, the bound $val(RP3)$ is equal to the optimal value of P3, referred to as $val(P3)$, while 4 of them have lower bounds with deviations of at most 4.72% from $val(P3)$.

4.3.3 Attaining Quick Lower and Upper Bounds

We propose two upper bounds UB1 and UB2 and two lower bounds LB1 and LB2 to restrict R. We obtain UB1 from the following 2-approximation algorithm for the p-center problem with $p \geq 2$. The algorithm constructs a set $X \subset N$ of centers with $|X| = p$ and allocates each vertex to its closest center. In order to construct X, the two most distant vertices in the network are initially added to the set. While X has less than p elements, the vertex that is most distant to X is added to the set. After allocating each vertex to the closest center in X, we obtain a feasible solution to the p-center problem. The objective value of this solution is no more than two times the optimal solution value [34]. We refer to this objective value as UB1.

Table 4.1: The lower bounds $val(RP3)$ and solution times of RP3 for OR-Library instances

Instance	n	p	$val(P3)$	$val(RP3)$	Gap (%)	Time (sec)
pmed1	100	5	127	121	4.72	0.19
pmed2	100	10	98	98	0	0.04
pmed3	100	10	93	93	0	0.03
pmed4	100	20	74	74	0	0.02
pmed5	100	33	48	48	0	0.02
pmed6	200	5	84	83	1.19	0.07
pmed7	200	10	64	64	0	0.09
pmed8	200	20	55	55	0	0.05
pmed9	200	40	37	37	0	0.04
pmed10	200	67	20	20	0	0.03
pmed11	300	5	59	59	0	0.10
pmed12	300	10	51	51	0	0.12
pmed13	300	30	36	36	0	0.06
pmed14	300	60	26	26	0	0.08
pmed15	300	100	18	18	0	0.02
pmed16	400	5	47	47	0	0.14
pmed17	400	10	39	39	0	0.17
pmed18	400	40	28	28	0	0.10
pmed19	400	80	18	18	0	0.06
pmed20	400	133	13	13	0	0.04
pmed21	500	5	40	40	0	0.20
pmed22	500	10	38	38	0	0.33
pmed23	500	50	22	22	0	0.16
pmed24	500	100	15	15	0	0.09
pmed25	500	167	11	11	0	0.05
pmed26	600	5	38	37	2.63	0.38
pmed27	600	10	32	32	0	0.34
pmed28	600	60	18	18	0	0.2
pmed29	600	120	13	13	0	0.14
pmed30	600	200	9	9	0	0.09
pmed31	700	5	30	30	0	0.31
pmed32	700	10	29	28	3.45	0.50
pmed33	700	70	15	15	0	0.28
pmed34	700	140	11	11	0	0.13
pmed35	800	5	30	30	0	0.46
pmed36	800	10	27	27	0	0.57
pmed37	800	80	15	15	0	0.44
pmed38	900	5	29	29	0	0.68
pmed39	900	10	23	23	0	0.93
pmed40	900	90	13	13	0	0.38
Average:						0.20

We obtain UB2 by making an improvement on the above algorithm. Let $\{x_1, \ldots, x_p\}$ be the set of centers selected. We divide the network into p clusters $I_1, \ldots, I_p$ where I_j is the set of vertices allocated to center x_i (break ties arbitrarily). Let $T_i = \max\{d(x_i, j) : j \in I_i\}$ for $i \in \{1, \ldots, p\}$. Then, the objective value of the current solution is $T_{max} = \max_{i=1,\ldots,p} T_i$. In order to improve the obtained solution, we solve a 1-center problem in each of these clusters by using the method proposed in [4]. We start with cluster I_i, where $T_i = T_{max}$ and obtain a new radius value $\overline{T}_i \leq T_i$. If $\overline{T}_i = T_i$, we stop the algorithm. If $\overline{T}_i < T_i$, we set $T_{max} = \overline{T}_i$. Then we solve a 1-center problem for each remaining cluster I_k if $T_k > T_{max}$ and set $T_{max} = \overline{T}_k$ if $= \overline{T}_k > T_{max}$. Obviously, we will have a solution which is no worse than the initial one after this improvement procedure. Thus, we can conclude that our algorithm is a 2-approximation algorithm. We refer to this solution value as UB2.

LB1 is obtained as follows: Suppose we sort positive distance values in non-decreasing order as $\beta_1 \leq \beta_2 \leq \ldots \leq \beta_{n\times(n-1)}$ (ties are allowed in the sequence) and let $X = \{x_1, x_2, \ldots, x_p\} \subset N$ be an optimal p-center. Then the remaining $n - p$ vertices need to be served by these centers. Even if we assume that each vertex is served by its closest center, the maximum of the closest distance values cannot be smaller than β_{n-p} since $val(P3) = \max_{i\in N\setminus X} \min_{x_j\in X} d_{ix_j} \geq \beta_{n-p}$. We refer to this lower bound as LB1.

LB2 is obtained from UB1: Since UB1 is less than or equal to two times the optimal value, (UB1)/2 provides a lower bound on the optimal value. We improve this lower bound one more step and select the smallest distance value in R, which is greater than or equal to (UB1)/2, as a lower bound (LB2) for the p-center problem.

We compute the LB1, LB2, UB1 and UB2 values for the 40 p-median instances and give the results obtained in Table 4.2 and Table 4.3. The gap values reported in Table 4.2 and Table 4.3 are equal to (Value$-val(P3))/val(P3)$ and $(val(P3)-$Value$)/val(P3)$, respectively, where 'Value' represents the corresponding bound value. When we compare the values of UB1 and UB2, we see that in 23 of these instances, UB2 value is smaller than UB1 value. This means that

the improvement stage is helpful in obtaining a solution with better objective in these instances. In the other instances, the improvement stage is not able to find a solution with a smaller objective value; thus, UB2 value equals UB1 value. When the values of LB1 and LB2 are compared, we observe that LB2 value is larger than LB1 value for each instance. Each of these lower and upper bounds is obtained in at most 0.06 seconds, which is much faster than the calculation times of any relaxation bound discussed. When we compare LB2 with $val(LP4)$ and $val(RP3)$, we see that LB2 and $val(LP4)$ values are quite close to each other and $val(RP3)$ is greater than both of them for each instance. Since we can obtain LB2 values very quickly, we decided to use LB2 and UB2 values to restrict the set R when we make experiments with the proposed model P3 and double bound algorithms. For any problem tested in this chapter, LB2 and UB2 values can be obtained in less than 1 second. Therefore, we do not add the calculation times of these bounds to the solution times reported in the tables of the remaining sections of this chapter.

4.4 Double Bound Algorithms

Let S be any nonempty subset of $T = \{1, \ldots, M\}$ and define $P(S)$ to be the problem which is exactly the same as P3 except that all variables $z_k, k \in T \setminus S$, are dropped from P3. This amounts to replacing the index set T in (4.12)-(4.15) with the index set S. Let $val(P(S))$ be the optimal value of $P(S)$ with $P(S) = \infty$ if $P(S)$ is infeasible.

Proposition 2. *Suppose $|S| > 2$. Let a and b be the smallest and largest indices in S, respectively.*

(a) If $val(P(S)) = \rho_a$, then $r_V^* \in \{\rho_1, \ldots, \rho_a\}$.

(b) If $val(P(S)) = \rho_k$ for some k with $a < k \leq b$, then $r_V^* \in \{\rho_{k'+1}, \ldots, \rho_k\}$ where k' is the largest index in S which is smaller than k.

(c) If $val(P(S)) = \infty$, then $r_V^* \in \{\rho_{b+1}, \ldots, \rho_M\}$.

Table 4.2: Values, gaps, and calculation times (seconds) of UB1 and UB2 for OR-Library instances

				UB1			UB2		
Instance	n	p	$val(P3)$	Value	Gap	Time	Value	Gap	Time
pmed1	100	5	127	202	0.59	0	191	0.50	0
pmed2	100	10	98	166	0.69	0.02	155	0.58	0.02
pmed3	100	10	93	145	0.56	0	143	0.54	0
pmed4	100	20	74	112	0.51	0	92	0.24	0
pmed5	100	33	48	75	0.56	0.02	75	0.56	0.02
pmed6	200	5	84	138	0.64	0	107	0.27	0.02
pmed7	200	10	64	102	0.59	0	102	0.59	0
pmed8	200	20	55	83	0.51	0	83	0.51	0
pmed9	200	40	37	53	0.43	0	53	0.43	0
pmed10	200	67	20	31	0.55	0	31	0.55	0
pmed11	300	5	59	98	0.66	0	69	0.17	0
pmed12	300	10	51	92	0.80	0	76	0.49	0
pmed13	300	30	36	59	0.64	0	55	0.53	0
pmed14	300	60	26	40	0.54	0	39	0.50	0
pmed15	300	100	18	24	0.33	0	24	0.33	0
pmed16	400	5	47	86	0.83	0	56	0.19	0
pmed17	400	10	39	69	0.77	0	57	0.46	0
pmed18	400	40	28	46	0.64	0	46	0.64	0
pmed19	400	80	18	29	0.61	0	27	0.50	0
pmed20	400	133	13	18	0.38	0.02	18	0.38	0.02
pmed21	500	5	40	71	0.78	0	49	0.23	0
pmed22	500	10	38	63	0.66	0	54	0.42	0
pmed23	500	50	22	35	0.59	0	35	0.59	0
pmed24	500	100	15	23	0.53	0.02	23	0.53	0.02
pmed25	500	167	11	15	0.36	0.03	15	0.36	0.03
pmed26	600	5	38	65	0.71	0	52	0.37	0
pmed27	600	10	32	57	0.78	0	46	0.44	0
pmed28	600	60	18	30	0.67	0	29	0.61	0
pmed29	600	120	13	21	0.62	0.03	21	0.62	0.03
pmed30	600	200	9	13	0.44	0.06	13	0.44	0.06
pmed31	700	5	30	54	0.80	0	36	0.20	0
pmed32	700	10	29	50	0.72	0.02	38	0.31	0.02
pmed33	700	70	15	25	0.67	0	25	0.67	0
pmed34	700	140	11	17	0.55	0.05	17	0.55	0.05
pmed35	800	5	30	53	0.77	0	34	0.13	0
pmed36	800	10	27	49	0.81	0	36	0.33	0
pmed37	800	80	15	26	0.73	0.03	26	0.73	0.03
pmed38	900	5	29	51	0.76	0	32	0.10	0
pmed39	900	10	23	39	0.70	0	30	0.30	0.02
pmed40	900	90	13	21	0.62	0.03	21	0.62	0.03
Average:					0.63	0.01		0.44	0.01

Table 4.3: Values, gaps, and calculation times (seconds) of LB1 and LB2 for OR-Library instances

				LB1			LB2		
Instance	n	p	$val(P3)$	Value	Gap	Time	Value	Gap	Time
pmed1	100	5	127	59	0.54	0	101	0.20	0
pmed2	100	10	98	56	0.43	0	83	0.15	0.02
pmed3	100	10	93	55	0.41	0	73	0.22	0
pmed4	100	20	74	41	0.45	0	56	0.24	0
pmed5	100	33	48	23	0.52	0	38	0.21	0.02
pmed6	200	5	84	38	0.55	0	69	0.18	0
pmed7	200	10	64	34	0.47	0	51	0.20	0
pmed8	200	20	55	30	0.45	0	42	0.24	0
pmed9	200	40	37	22	0.41	0	27	0.27	0
pmed10	200	67	20	11	0.45	0	16	0.20	0
pmed11	300	5	59	34	0.42	0	49	0.17	0
pmed12	300	10	51	30	0.41	0	46	0.10	0
pmed13	300	30	36	20	0.44	0.02	30	0.17	0
pmed14	300	60	26	14	0.46	0	20	0.23	0
pmed15	300	100	18	10	0.44	0	12	0.33	0
pmed16	400	5	47	26	0.45	0	43	0.09	0
pmed17	400	10	39	21	0.46	0	35	0.10	0
pmed18	400	40	28	16	0.43	0	23	0.18	0
pmed19	400	80	18	10	0.44	0	15	0.17	0
pmed20	400	133	13	7	0.46	0	9	0.31	0.02
pmed21	500	5	40	23	0.43	0.02	36	0.10	0
pmed22	500	10	38	21	0.45	0	32	0.16	0
pmed23	500	50	22	13	0.41	0	18	0.18	0
pmed24	500	100	15	9	0.40	0	12	0.20	0.02
pmed25	500	167	11	6	0.45	0	8	0.27	0.03
pmed26	600	5	38	21	0.45	0	33	0.13	0
pmed27	600	10	32	18	0.44	0	29	0.09	0
pmed28	600	60	18	10	0.44	0.02	15	0.17	0
pmed29	600	120	13	7	0.46	0	11	0.15	0.03
pmed30	600	200	9	5	0.44	0	7	0.22	0.06
pmed31	700	5	30	16	0.47	0	27	0.10	0
pmed32	700	10	29	16	0.45	0.02	25	0.14	0.02
pmed33	700	70	15	9	0.40	0.02	13	0.13	0
pmed34	700	140	11	6	0.45	0.02	9	0.18	0.05
pmed35	800	5	30	16	0.47	0.02	27	0.10	0
pmed36	800	10	27	16	0.41	0	25	0.07	0
pmed37	800	80	15	8	0.47	0.02	13	0.13	0.03
pmed38	900	5	29	15	0.48	0.02	26	0.10	0
pmed39	900	10	23	13	0.43	0.02	20	0.13	0
pmed40	900	90	13	7	0.46	0.02	11	0.15	0.03
Average:					0.45	0.01		0.17	0.01

Proof. Since S is a subset of T, $P(S)$ is a restriction of P3. Hence, $val(P3) \leq val(P(S))$. We have $r_V^* = val(P3)$ from Proposition 1 implying that $r_V^* \leq \rho_a$ if (a) holds. This proves (a). To prove (b), observe that $val(P(S)) = \rho_k$ and $k > a$ imply that P_k is feasible while $P_{k'}$ is infeasible. Accordingly, $\rho_{k'} < r_V^* \leq \rho_k$ which gives $r_V^* \in \{\rho_{k'+1}, \ldots, \rho_k\}$. To prove (c), observe that $val(P(S)) = \infty$ implies that P_b is infeasible. Hence, $r_V^* \in \{\rho_{b+1}, \ldots, \rho_M\}$. □

This proposition allows devising a search strategy based on restrictions of P3 or P4. The main idea is the following: Select a nonempty subset S of T and solve $P(S)$. Depending on which of (a), (b) or (c) occurs in Proposition 2, delete from T the set of indices k such that ρ_k cannot equal r_V^*. Select a new subset S of T after deletions and repeat the procedure. Termination occurs when T reduces to a singleton. The computational success of such a procedure depends on how efficiently we solve each subproblem, $P(S)$, as well as how many times we have to solve a new problem. We give below a specialized way of doing this with sets S containing two elements. The method is called the double bound method. Six variations are discussed.

The double bound algorithm solves $P(S)$ for $S = \{a, b\}$ where $a, b \in T$ with $a < b$. Let $T_1 = \{1, \ldots, a\}, T_2 = \{a + 1, \ldots, b\}$ and $T_3 = \{b + 1, \ldots, M\}$. We may solve $P(S)$ directly or solve P_a and P_b separately. The former problem involves $m + 2$ binary variables while each of the latter problems involves m binary variables. After solving $P(S)$, we know which one of the subsets T_1, T_2 and T_3 contains the optimal value of the p-center problem. Since we do not need to consider two of these subsets anymore, we repeat the procedure with the subset that contains the optimal value until we have a single element subset on hand. We propose three ways to choose a and b in Table 4.4. For each choice, we give two types of algorithms, one type solving $P(S)$ directly (referred to as DB algorithms) and the other solving P_a and P_b separately to obtain a solution to $P(S)$ (referred to as DBR algorithms). The initial steps and the terminations of the algorithms are the same. The algorithms work with the ordered list $R = \{\rho_1, \ldots, \rho_M\}$. We provide a general scheme of the double bound algorithm in Figure 1. Note that the DBR algorithms are identical for solving P3 and P4.

Table 4.4: Selection of radius values for the double bound algorithm

(a, b)	Together	Separately
$a = \lfloor(\max + \min)/2\rfloor$ $b = \max$	DB1	DBR1
$a = \lfloor(\max + \min)/2\rfloor$ $b = \max - 1$	DB2	DBR2
$a = \min + \lfloor(\max - \min)/3\rfloor$ $b = \min + 2\lfloor(\max - \min)/3\rfloor$	DB3	DBR3

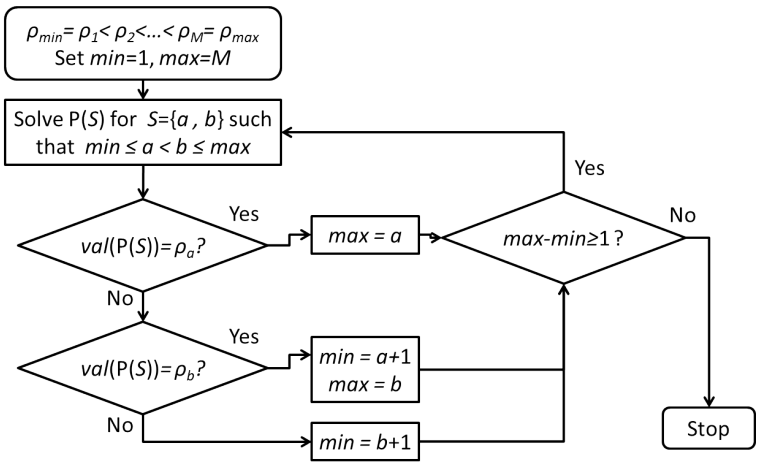


Figure 4.1: General scheme of the double bound algorithm

In the worst case, DB1, DBR1, DB2 and DBR2 algorithms terminate in $O(log_2 M)$ iterations and DB3, DBR3 algorithms terminate in $O(log_3 M)$ iterations. Suppose we are at the beginning of some iteration of our DB2 algorithm and we have an a value and a max value. Then we solve $P(S)$ for $S = \{a, \max -1\}$. If r_V^* equals $\rho_{\max}$ of this iteration, then $P(S)$ becomes infeasible and we terminate the algorithm at the end of this iteration. This quick termination is also available in DBR2 algorithm. In DB1 algorithm, we solve $P(S)$ for $S = \{a, \max\}$. If r_V^* equals $\rho_{\max}$ of the current iteration, we cannot know this until we solve $P(S)$ for $S = \{\max -1, \max\}$ and this takes $O(log_2(\max -a))$ iterations. This is also valid for DBR1 algorithm. In the same case, DB3 and DBR3 algorithms terminate after $O(log_3(\max -b))$ iterations.

4.5 Computational Experiments

In the computational experiments of this chapter, we follow the general trend in the literature and take $J = N$. The input data used for the computations consists of the p-median data from OR-Library [15] for 40 instances with n varying between 100 and 900 and p varying between 5 and $(n/3)$ and some instances from TSPLIB [16]. The original data in OR-Library [15] consists of a listing of edges and their lengths. By using the all-pairs shortest path algorithm due to Floyd [62] on this data, we obtain the distance matrix D. In TSPLIB [16] instances, the coordinates of the vertices are provided. We calculate the Euclidean distance for each vertex pair and round it to the nearest integer. Note that rounding might destroy the triangular inequality, but we are safe since we do not require this property in our methods. We solve problems from u1817 with $n = 1817$, from d15112 with $n = 2500$ and from pcb3038 with $n = 3038$. During our computations we use JCreator LE 4.50 [63] and IBM ILOG CPLEX 12.4 [64] with concert technology. In all mathematical models that we solve in this chapter, we set MIPEmphasis option of CPLEX to 1. For the algorithms DBR1, DBR2 and DBR3, we additionally set IntSolLim to 1.

In all tables of this section, the first three columns give the characteristics

of the instances and the column labeled "Opt" gives the optimal p-radii. The reported solution times in tables are in seconds. We provide the results of the experiments on the unweighted problems in Section 4.5.1 and the weighted problems in Section 4.5.2.

4.5.1 Unweighted Problems

While P1 is a more compact formulation of the p-center problem than P2, P3 and P4, the computational performances of P2, P3 and P4 are far better than that of P1 when all four formulations are directly solved by a commercial solver. Table 4.5 gives a comparison of solution times and optimal or best-found objective values for P1, P2, P3 and P4 for the vertex restricted case using 40 p-median instances taken from the OR-Library [15]. The possible values of r_V^* in R that are needed in P2, P3 and P4 are restricted to a subset $R' \equiv R \cap [LB2, UB2]$. In this table, columns 5, 6, 7 and 8 give the solution times in seconds taken by the solver for the IP models P4, P3, P2 and P1, respectively. While solving the four IP models, we put a time restriction of three hours. P2, P3 and P4 can solve each problem optimally within the three hour limit while P1 cannot solve 17 of these problems optimally. Of the 17 unsolved instances, 5 of them are instances for which no feasible integer solution can be found by P1 within the three hour limit. These are indicated by NFS (No Feasible Solution) in the middle column under P1 in the table. For the 12 instances that are solved sub-optimally by P1, the last column under P1 gives the percent gap reported by CPLEX. When we compare P2, P3 and P4 with each other, we see that the largest time required is 772.68 seconds for P4 (instance pmed39) while it is 1232.92 seconds for P3 (instance pmed39) and 1428.94 seconds for P2 (instance pmed8). The average time required to solve 40 instances with P4 is 82.25 seconds while it is 99.80 seconds for P3 and 155.25 seconds for P2. The differences in computational times are more significant in some instances. For example, P3 solves pmed8 in 10.31 seconds and P4 solves the same instance in 12.74 seconds while P2 solves it in 1428.94 seconds, which is more than 138 times that of P3 and 112 times that of P4. Among 40 instances, for 7 of them P2 achieves the smallest solving time while P3 is the successor for

22 of them and P4 is the successor for 11 of them. We may conclude from these observations that the proposed model P3 and its tightened version P4 perform noticeably better than P2 on the 40 p-median instances taken from OR-Library [15] and that P2, P3 and P4 perform significantly better than P1. While direct solution times are reasonable for P2, P3 and P4, the double bound algorithms lead to much shorter solution times than direct solution times available in this table.

We compare the performance of our algorithms on 40 p-median instances in Table 4.6. The first observation from this table is that the double bound algorithms perform much better than solving P3 or P4 directly (See Table 4.5.). The solution times of the double bound algorithms are much better for all instances than direct solution times. When we compare the double bound algorithms among themselves, we see that they all show similar performance in terms of time requirements for the instances with n =100 or 200. As n goes up from 200 to 900, the DBR algorithms take considerably less time to terminate than DB algorithms. This is also reflected in the average time requirements of the algorithms. Each instance in this table is solved in less than 3 seconds by DBR algorithms and in less than 27 seconds by DB algorithms.

To see the computational performance of the proposed algorithms on larger problems, we conducted experiments on u1817 data from TSPLIB [16]. This problem has 1817 nodes and is solved with 18 different p values. We observed in the tests that the DB algorithms require significantly more time to arrive at optimal solutions than DBR algorithms. For this reason, we give the results only for the DBR algorithms.

Table 4.7 gives the solution times for the DBR algorithms on 20 instances from TSP-Library with n=1817. We again restrict R by using LB2 and UB2 values in these experiments and these values are given in the fifth and sixth columns, respectively. The solution times of the algorithms are given in the last three columns. For the highlighted instances in this table, the optimal values are provided for the first time in the literature. Elloumi et al. [1] conduct experiments on the first 15 instances of this table, but they are not able to solve optimally

Table 4.5: Solution times (seconds) of IP models

Instance	n	p	Opt	P4 Time	P3 Time	P2 Time	P1 Time	P1 Obj	Gap%
pmed1	100	5	127	11.33	4.38	52.78	188.81	127	
pmed2	100	10	98	7.0	2.00	12.94	73.34	98	
pmed3	100	10	93	5.26	1.90	7.72	37.89	93	
pmed4	100	20	74	1.30	0.21	8.35	0.90	74	
pmed5	100	33	48	1.08	0.26	1.45	0.39	48	
pmed6	200	5	84	11.50	15.81	33.50	2797.12	84	
pmed7	200	10	64	26.89	16.67	96.79	4476.78	64	
pmed8	200	20	55	12.74	10.31	1428.94	776.21	55	
pmed9	200	40	37	4.71	0.68	3.57	6.40	37	
pmed10	200	67	20	0.62	0.13	0.34	1.40	20	
pmed11	300	5	59	9.09	15.08	14.01	537.31	59	
pmed12	300	10	51	34.74	38.42	28.93	7115.81	51	
pmed13	300	30	36	16.05	19.83	42.15	3188.15	36	
pmed14	300	60	26	5.60	2.19	4.92	281.05	26	
pmed15	300	100	18	1.33	0.29	0.81	3.82	18	
pmed16	400	5	47	5.76	4.18	33.91	10800.00	47	6.78
pmed17	400	10	39	45.21	50.32	38.33	10800.00	41	14.63
pmed18	400	40	28	71.64	48.98	55.39	10539.49	28	
pmed19	400	80	18	4.91	2.24	5.43	33.34	18	
pmed20	400	133	13	1.33	0.28	0.75	3.79	13	
pmed21	500	5	40	12.36	17.95	35.65	10800.00	43	16.28
pmed22	500	10	38	239.84	373.73	1223.58	10800.00	41	18.37
pmed23	500	50	22	185.59	54.86	129.87	10800.00	24	16.67
pmed24	500	100	15	10.33	3.46	4.78	439.70	15	
pmed25	500	167	11	1.61	0.26	0.73	17.13	11	
pmed26	600	5	38	32.64	72.78	57.96	10800.00	39	10.26
pmed27	600	10	32	95.41	295.62	201.31	10800.00	40	25.00
pmed28	600	60	18	180.21	75.00	46.47	10800.00	20	19.72
pmed29	600	120	13	19.19	7.24	12.28	1976.99	13	
pmed30	600	200	9	1.25	0.31	0.70	5.29	9	
pmed31	700	5	30	19.75	15.70	34.13	10800.00	NFS	
pmed32	700	10	29	351.59	407.49	949.57	10800.00	38	32.89
pmed33	700	70	15	177.61	180.43	213.66	10800.00	17	17.46
pmed34	700	140	11	22.98	6.91	7.59	3159.30	11	
pmed35	800	5	30	18.38	14.53	14.25	10800.00	NFS	
pmed36	800	10	27	358.30	414.04	227.85	10800.00	NFS	
pmed37	800	80	15	186.11	268.12	154.61	10800.00	25	47.64
pmed38	900	5	29	17.64	20.63	19.53	10800.00	NFS	
pmed39	900	10	23	772.68	1232.82	837.51	10800.00	NFS	
pmed40	900	90	13	308.41	295.98	167.11	10800.00	20	40.00
Average:				82.25	99.80	155.25	5481.51		

Table 4.6: Times (seconds) required to solve P3 with DB and DBR algorithms on 40 p-median instances

Instance	n	p	DB1	DB2	DB3	DBR1	DBR2	DBR3
pmed1	100	5	0.41	0.58	0.55	0.46	0.54	0.54
pmed2	100	10	0.17	0.15	0.31	0.19	0.23	0.23
pmed3	100	10	0.19	0.13	0.18	0.20	0.21	0.12
pmed4	100	20	0.19	0.18	0.41	0.11	0.13	0.21
pmed5	100	33	0.13	0.18	0.11	0.10	0.12	0.09
pmed6	200	5	0.68	0.80	0.60	0.29	0.32	0.23
pmed7	200	10	0.38	0.32	0.48	0.18	0.18	0.20
pmed8	200	20	0.26	0.37	0.49	0.14	0.19	0.17
pmed9	200	40	0.16	0.27	0.23	0.12	0.20	0.20
pmed10	200	67	0.10	0.06	0.27	0.09	0.10	0.06
pmed11	300	5	0.63	0.65	0.50	0.43	0.54	0.31
pmed12	300	10	0.79	0.71	0.96	0.28	0.32	0.30
pmed13	300	30	0.44	0.45	0.66	0.26	0.30	0.27
pmed14	300	60	0.30	0.30	0.37	0.12	0.15	0.13
pmed15	300	100	0.11	0.11	0.30	0.05	0.07	0.09
pmed16	400	5	1.12	0.91	1.00	0.44	0.55	0.50
pmed17	400	10	1.43	0.78	1.12	0.43	0.33	0.38
pmed18	400	40	0.93	0.82	0.87	0.15	0.17	0.14
pmed19	400	80	0.39	0.38	0.37	0.11	0.11	0.09
pmed20	400	133	0.17	0.13	0.13	0.05	0.05	0.06
pmed21	500	5	1.99	1.66	1.49	0.81	0.95	0.92
pmed22	500	10	3.07	3.26	3.43	1.06	1.08	0.99
pmed23	500	50	1.42	1.41	1.67	0.22	0.31	0.29
pmed24	500	100	0.58	0.42	0.24	0.14	0.14	0.09
pmed25	500	167	0.26	0.18	0.17	0.07	0.07	0.07
pmed26	600	5	5.68	6.25	4.31	1.62	1.61	1.15
pmed27	600	10	4.23	3.56	3.29	0.98	0.94	0.89
pmed28	600	60	2.34	2.05	1.73	0.27	0.29	0.24
pmed29	600	120	0.76	0.87	0.45	0.14	0.18	0.14
pmed30	600	200	0.32	0.24	0.28	0.09	0.08	0.13
pmed31	700	5	4.20	5.22	2.51	1.32	2.03	1.22
pmed32	700	10	6.62	5.52	7.39	1.91	2.35	2.12
pmed33	700	70	3.55	2.69	1.60	0.37	0.33	0.28
pmed34	700	140	0.90	0.50	0.70	0.16	0.15	0.22
pmed35	800	5	6.88	4.42	5.29	1.72	1.69	1.72
pmed36	800	10	12.86	7.30	9.82	2.14	2.71	1.96
pmed37	800	80	6.38	4.02	3.01	0.46	0.40	0.31
pmed38	900	5	11.44	6.63	6.67	2.53	2.55	2.61
pmed39	900	10	12.00	26.50	23.56	2.69	2.59	1.57
pmed40	900	90	7.59	6.29	3.37	0.47	0.61	0.44
Average:			2.55	2.43	2.27	0.58	0.65	0.54

Table 4.7: Results of DBR algorithms for solving P3 or P4 for TSPLIB instances with n=1817

Instance	n	p	Opt	LB2	UB2	DBR1	DBR2	DBR3
u1817	1817	10	458	301	585	27.92	22.11	25.45
u1817	1817	20	**309**	191	357	305.45	278.49	452.52
u1817	1817	30	**241**	166	331	274.13	344.59	147.97
u1817	1817	40	**209**	155	307	789.89	1221.86	896.55
u1817	1817	50	**185**	127	229	332.89	330.09	643.78
u1817	1817	60	163	105	209	16.38	17.52	13.14
u1817	1817	70	148	99	194	6.02	6.04	9.93
u1817	1817	80	137	93	185	26.79	35.12	24.95
u1817	1817	90	**129**	90	180	8619.10	7519.04	9545.52
u1817	1817	100	127	86	170	7.26	10.22	12.6
u1817	1817	110	110	81	161	4.87	5.32	5.33
u1817	1817	120	107	74	148	3.16	3.99	2.99
u1817	1817	130	**105**	71	137	337.57	335.03	373.23
u1817	1817	140	**102**	68	129	7.73	4.62	9.11
u1817	1817	150	**92**	64	127	6.52	6.61	6.73
u1817	1817	200	**80**	56	105	2.1	2.56	2.37
u1817	1817	250	**76**	46	92	1.26	2.17	1.42
u1817	1817	300	**63**	46	80	0.87	2.01	0.89
Average:						598.33	563.74	676.36

the highlighted instances. The algorithms spend an excessive amount of time for solving the problem with p=90. DBR2 shows the best performance and DBR3 shows the worst performance for this instance. Another instance that results in a considerable difference between the performances of the algorithms is the problem with p=40. DBR1 performs the best and DBR2 performs the worst for this instance. For the other instances, the time requirements of the algorithms are very close to each other. On the average, DBR2 spends the least amount of time and DBR3 spends the largest amount of time.

We select DBR2 as the winner of the six DB and DBR algorithms and solve larger problems with $n \in \{1817, 2500, 3008\}$ from the TSPLIB [16] by using the lower bound $val(RP3)$. We compute this bound by solving $O(log_2 M)$ LP problems RP_h using Algorithm 1 with R restricted to $R \cap [LB2, UB2]$. The bounds $val(RP3)$ are generally computed in a matter of seconds. The minimum, average and maximum computing times for 33 instances are 0.59, 33.16 and 166

seconds, respectively. We also note that the lower bound $val(RP3)$ is equal to the optimal value of P3 for 8 instances of the TSPLIB [16] problems with $n \in \{1817, 2500, 3008\}$ while the gap between them is at most 0.053 for the remaining instances.

Table 4.8 gives the relevant statistics for solving P3 via the algorithm DBR2 by restricting R to $R \cap [val(RP3), UB2]$. For the 33 instances reported in Table 4.8, a time limit of three hours is imposed for each problem $P_h \in \{P_{mid}, P_{\max -1}\}$ attempted in main steps of the algorithm DBR2. At the end of three hours, either a solution is found for P_h or the infeasibility of P_h is detected or the feasibility/infeasibility status of P_h remains unknown. If the status of P_h remains unknown, we solve the problems P_{h+1}, P_{h+2}, etc. with successively increasing radius values until a feasible solution is found within three hours for some problem P_{h+k}. This gives us an upper bound ρ_{h+k} on r_V^* that is the best (smallest) bound that can be confirmed within the time limit. Similarly, we solve problems P_{h-1}, P_{h-2}, etc. with successively decreasing radius values until we detect infeasibility of some $P_{h-k'}$ within the time limit. This gives us a lower bound $\rho_{h-k'+1}$ on r_V^* that is the largest possible that can be confirmed within the time limit. Thus, $r_V^* \in \{\rho_{h-k'+1}, \ldots, \rho_{h+k}\}$. The values $\rho_{h-k'+1}$ and ρ_{h+k} are reported as the "Best LB" and "Best UB" in the table in columns 5 and 6, respectively. Column 7 gives the gap between the "Best UB" and "Best LB" values while column 4 gives the optimal values for P3. If the optimal solution is found for a problem within three hours, the values in columns 4, 5 and 6 are equal. For such problems, the gaps in column 7 are zero. Column 8 gives the solution times obtained by the algorithm DBR2. For all of the instances with $n = 1817$, we obtain the optimal solution. The most time consuming instance among them is the one with $p = 40$. This instance is solved in 1225 seconds. For the problems with $n = 2500$, we obtain the optimal solution for all instances except the one with $p = 100$. The gap between the best upper bound and the best lower bound we obtain for this instance is 0.008. The optimal solutions for the six instances with $n = 3038$ are obtained. For the remaining instances the gap between the best bounds is at most 0.027. For any of the reported network sizes, all instances with $p = 5$ and 10 are solved in less than three minutes and all instances with $p = 400$ and 500

are solved in less than 12 seconds.

Before solving the individual problem P_k for some $k \in T$ in the main steps of DBR2 algorithm, it is possible to eliminate some of the variables and constraints by applying the reduction rules which are well known for the set covering problem [65]. These rules not only detect any infeasibility immediately, but also reveal the optimal value of some decision variables and the dominance relation between the constraints. We utilized these reduction rules in our algorithm to solve the instances that require large amount of time (more than one hour) and provided the results in Table 4.9. One of the instances in this table is taken from Table 4.7 and the others are taken from Table 4.8. Table 4.9 consists of the same columns as in Table 4.8 except that it has an additional column "PP Time" which represents the total time consumed for the reduction procedure. Among the 13 instances in this table, for 11 of them reduction makes an improvement in terms of either the best LB, the best UB or in the total time and those improved parameters are depicted in bold. We observe that the maximum gap between the best LB and the best UB decreases from 0.027 to 0.022 and we are able to obtain the optimal solution for the instance with $n = 3038$ and $p = 30$. For some instances the total time consumption decreases significantly. For instance, the total time consumed decreases around 85% for the first instance and 84% for the second instance in Table 4.9. The average decrease in total time is 61% for the instances with decrease in time and 38% for all instances. When we subtract the reduction time from the total time, we can observe how much reduction helps CPLEX to solve the problems. The average of this improvement is 60% for the improved instances and 49% for all instances. These experiments reveal that the reduction rules are quite helpful in solving both individual problems P_k for $k \in T$ and P3 via DBR2 algorithm.

4.5.2 Weighted Problems

In this section, we solve weighted problems by using the DBR2 algorithm. We generate the weights of the nodes uniformly between 1 and 10. Table 4.10 presents the results we obtain from 40 OR-Library [15] instances. The average solution

Table 4.8: Results of algorithm DBR2 for solving problem P3 or P4 for TSPLIB instances with $n \in \{1817, 2500, 3038\}$

Instance	n	p	*val(P3)*	Best LB	BestUB	Gap	Time (sec)
u1817	1817	5	715	715	715	0	14.10
u1817	1817	10	458	458	458	0	20.31
u1817	1817	20	309	309	309	0	257.68
u1817	1817	30	241	241	241	0	204.77
u1817	1817	40	209	209	209	0	1225.00
u1817	1817	50	185	185	185	0	314.46
u1817	1817	100	127	127	127	0	7.43
u1817	1817	200	80	80	80	0	2.57
u1817	1817	300	63	63	63	0	1.19
u1817	1817	400	51	51	51	0	0.84
u1817	1817	500	51	51	51	0	0.23
d15112	2500	5	5856	5856	5856	0	95.91
d15112	2500	10	3705	3705	3705	0	84.85
d15112	2500	20	2573	2573	2573	0	288.12
d15112	2500	30	2029	2029	2029	0	5293.59
d15112	2500	40	1723	1723	1723	0	21899.83
d15112	2500	50	1524	1524	1524	0	5782.76
d15112	2500	100		1049	1057	0.008	94245.98
d15112	2500	200	723	723	723	0	28371.08
d15112	2500	300	571	571	571	0	38.39
d15112	2500	400	481	481	481	0	4.99
d15112	2500	500	424	424	424	0	4.84
pcb3038	3038	5	1064	1064	1064	0	116.74
pcb3038	3038	10	729	729	729	0	176.28
pcb3038	3038	20	493	493	493	0	22740.76
pcb3038	3038	30		391	397	0.015	76923.67
pcb3038	3038	40		331	337	0.018	72364.56
pcb3038	3038	50		292	300	0.027	91029.77
pcb3038	3038	100		207	209	0.010	27292.39
pcb3038	3038	200		138	141	0.022	33929.91
pcb3038	3038	300	115	115	115	0	6185.96
pcb3038	3038	400	97	97	97	0	11.48
pcb3038	3038	500	85	85	85	0	5.23

Table 4.9: Results with utilization of the reduction rules

Instance	n	p	*val(P3)*	Best LB	Best UB	Gap	PP Time (sec)	Total Time (sec)
u1817	1817	90	129	129	129	0	578.59	**1225.76**
d15112	2500	30	2029	2029	2029	0	108.25	**865.63**
d15112	2500	40	1723	1723	1723	0	132.37	**6831.56**
d15112	2500	50	1524	1524	1524	0	158.75	**1645.55**
d15112	2500	100		**1050**	1059	0.009	416.46	104013.04
d15112	2500	200	723	723	723	0	482.84	**5293.26**
pcb3038	3038	20	493	493	493	0	3102.33	**6800.68**
pcb3038	3038	30	**393**	**393**	**393**	**0**	4063.66	**52285.79**
pcb3038	3038	40		**332**	337	**0.015**	4920.76	75702.72
pcb3038	3038	50		**293**	**299**	**0.020**	6199.33	**80313.04**
pcb3038	3038	100		207	**208**	**0.005**	4483.28	**15717.50**
pcb3038	3038	200		138	141	0.022	6060.09	42621.69
pcb3038	3038	300	115	115	115	0	3701.42	6569.00

time in this table is slightly larger than the average solution time obtained from solving the unweighted problems in Table 4.6. However, when we look at Tables 4.11 and 4.12, we can observe that the weighted problems are solved faster when compared to the solution times of the unweighted problems in Tables 4.7 and 4.8. This is an expected result since breaking symmetry typically eases a problem.

4.6 Conclusion

In this chapter, we proposed a new IP formulation for the p-center problem. By tightening one set of constraints in our model, we obtained a modified model with much better LP bounds. When compared with the two models from the literature, both of our models performed better in terms of the time requirement. The LP bounds of our tightened model are equivalent to the LP bounds of the model proposed by [1] and they are the strongest LP bounds among those of the four models. We obtained a stronger lower bound from our models by relaxing one set of binary variables and this lower bound is equivalent to the best known lower bound in the literature. We also provided a polynomial time algorithm to obtain this lower bound. In addition to the relaxation bounds, we proposed new lower and upper bounds which can be computed very quickly and we used

Table 4.10: Results for solving P3 or P4 with DBR2 algorithm on weighted OR-Library instances

Instance	n	p	Opt	Time (sec)
pmed1	100	5	2292	0.44
pmed2	100	10	1605	0.60
pmed3	100	10	1539	0.28
pmed4	100	20	1176	0.40
pmed5	100	33	664	0.05
pmed6	200	5	1410	0.52
pmed7	200	10	1156	0.44
pmed8	200	20	882	0.35
pmed9	200	40	602	0.28
pmed10	200	67	242	0.14
pmed11	300	5	1025	0.42
pmed12	300	10	928	0.41
pmed13	300	30	553	0.33
pmed14	300	60	369	0.25
pmed15	300	100	201	0.24
pmed16	400	5	817	0.61
pmed17	400	10	704	0.90
pmed18	400	40	465	0.29
pmed19	400	80	279	0.27
pmed20	400	133	171	0.22
pmed21	500	5	737	0.95
pmed22	500	10	686	1.28
pmed23	500	50	363	0.53
pmed24	500	100	228	0.39
pmed25	500	167	152	0.27
pmed26	600	5	722	1.31
pmed27	600	10	551	1.33
pmed28	600	60	295	0.53
pmed29	600	120	200	0.35
pmed30	600	200	118	0.24
pmed31	700	5	602	2.34
pmed32	700	10	552	1.73
pmed33	700	70	252	0.53
pmed34	700	140	168	0.36
pmed35	800	5	555	2.86
pmed36	800	10	516	3.12
pmed37	800	80	253	0.70
pmed38	900	5	527	3.26
pmed39	900	10	442	2.81
pmed40	900	90	221	0.96
Average				1.16

Table 4.11: Results for solving P3 or P4 with DBR2 algorithm on weighted TSPLIB instances with $n = 1817$

Instance	n	p	Opt	Time (sec)
u1817	1817	10	4161	14.85
u1817	1817	20	2701	9.02
u1817	1817	30	2051	21.51
u1817	1817	40	1781	14.16
u1817	1817	50	1521	16.69
u1817	1817	60	1301	5.60
u1817	1817	70	1234	7.21
u1817	1817	80	1141	5.56
u1817	1817	90	1011	5.63
u1817	1817	100	921	5.18
u1817	1817	110	890	5.30
u1817	1817	120	820	5.36
u1817	1817	130	761	3.90
u1817	1817	140	761	3.37
u1817	1817	150	737	3.29
u1817	1817	200	640	2.48
u1817	1817	250	511	2.26
u1817	1817	300	460	2.31
Average:				7.43

Table 4.12: Results for solving P3 or P4 with DBR2 algorithm on weighted TSPLIB instances with $n = 3038$

instance	n	p	Opt	Time (sec)
pcb3038	3038	5	9964	350.21
pcb3038	3038	10	6671	52.94
pcb3038	3038	20	4321	35.13
pcb3038	3038	30	3461	125.20
pcb3038	3038	40	2911	365.40
pcb3038	3038	50	2521	94.33
pcb3038	3038	100	1666	1181.89
pcb3038	3038	200	1081	5.07
pcb3038	3038	300	834	5.43
pcb3038	3038	400	694	6.22
pcb3038	3038	500	601	7.96
Average:				202.71

them effectively in our models and algorithms. We proposed a new method that solves successive restrictions of our model and we computationally tested a specialization of this algorithm, referred to as double bound algorithm, using benchmark problems from OR-Library [15] and TSPLIB [16]. We were able to solve problems with n=2500 and 3038 from the TSPLIB [16] using this algorithm while the largest problem solved in the literature had 1817 nodes. We solved the problems that require large amount of time by integrating the reduction rules to our algorithm. We observed significant improvements in utilization of the reduction rules. We conducted experiments on the weighted problems as well and we observed that especially larger weighted problems can be solved much faster when compared to the unweighted problems. [1]

[1] In this chapter, Figure 4.1, Tables 4.4, 4.5, 4.8, 4.9 and the related texts are reprinted from Computers & Operations Research, 40, H. Calik and B.C. Tansel, Double bound method for solving the p-center location problem, pp. 2991-2999, Copyright © 2013, with permission from Elsevier.

Chapter 5

Absolute p-Center Problem

In Chapter 4, we proposed new mathematical formulations and an algorithmic method to solve the p-center problem. However, the experimental studies we conducted in Chapter 4 were based on the vertex restricted problem. In this chapter, we focus on the absolute p-center problem on general networks. We implement the DBR2 algorithm proposed in Chapter 4.2 for solving the absolute p-center problem. In order to apply this method to the absolute p-center problem, construction of the finite set of potential facilities and distinct radius values is essential. We provide new upper and lower bounds for the problem and propose a method for generation of the set of potential facilities by utilizing the proposed lower and upper bounds. We make use of several theoretical results to decrease the number of potential facilities as much as possible so that we can solve the individual mathematical models in our method as quickly as possible. We solve problems with 900 demand nodes and 16056 edges in less than 509 seconds. Although the methods, discussions, and experiments are provided based on the unweighted version of the problem, they can be easily extended to weighted version of the problem as well.

This chapter is designed as follows: In Section 5.1, we present the details of our method to generate the finite set of potential facilities. In Section 5.2, we give improved lower and upper bounds for the absolute p-center problem. In Section 5.3, we provide the computational results of our experiments .

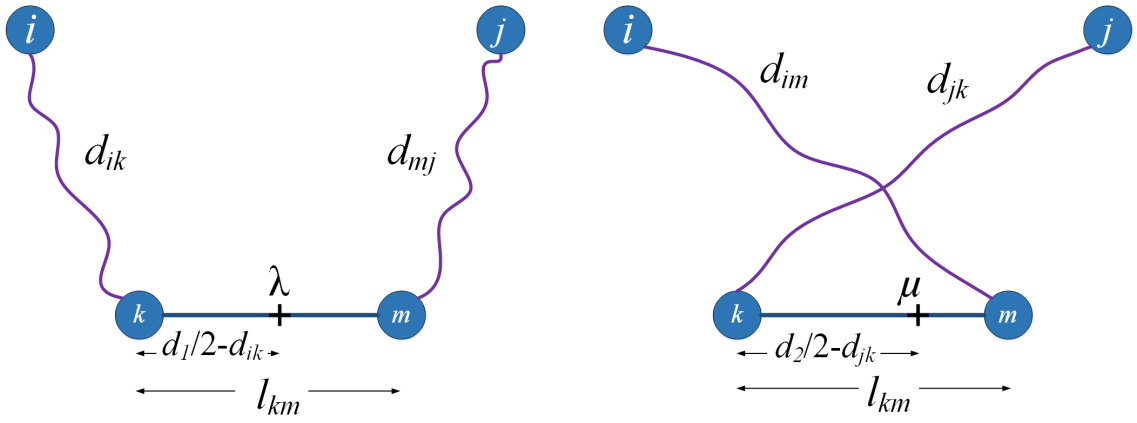


Figure 5.1: Illustration of intersection points on edge $\{k, m\}$ for node pair i, j.

5.1 Generation of the Intersection Points

The method we implement utilizes the fact that the set of potential facilities, which is denoted by J, in the absolute p-center problem can be restricted to $P \cup N$, where P is the set of intersection points. For this purpose, we develop a method for the generation of the intersection points.

In our method, we search the intersection points on each edge of the network. Let $\{k, m\}$ be an arbitrary edge in E. Now, we select a node pair $i, j \in N : i \neq j$. Then, we calculate the distance values $d_1 = d_{ik} + l_{km} + d_{jm}$ and $d_2 = d_{im} + l_{km} + d_{kj}$. Let λ and μ denote the midpoints of the paths associated with d_1 and d_2, respectively. In order to have an intersection point associated with node pair i, j on edge $\{k, m\}$, either λ or μ should fall on $\{k, m\}$. If both λ and μ are on edge $\{k, m\}$ and they are distinct points, then we select the midpoint of the path with smaller length as an intersection point, that is, if $d_1 < d_2$, we select λ; if $d_2 < d_1$ we select μ; if $d_1 = d_2$, we select neither of them [9]. If λ and μ are not distinct points, then we select $\lambda \equiv \mu$ as an intersection point. Therefore, we select at most one of λ or μ as an intersection point. See Figure 5.1 for an illustration of λ and μ on edge $\{k, m\}$ for node pair i, j. In this figure, λ is the unique point that satisfies $d_{ik} + d_{k\lambda} = d_{\lambda m} + d_{mj}$, and μ is the unique point that satisfies $d_{jk} + d_{k\mu} = d_{\mu m} + d_{mi}$.

The following properties, which are earlier used by Garfinkel et al. [9] as well, are effective in decreasing the number of intersections points and we integrate them to our method.

1. If $l_{km} > d_{km}$ for any $\{k, m\} \in E$, then we can disregard this edge for further considerations.

2. For an edge $\{k, m\} \in E$ and node pair $i, j \in N$, we have at most one intersection point in P.

In addition to the above properties, we reveal new properties to utilize in our method below: Suppose we have an upper bound value UB and a lower bound value LB for r_A^*.

Theorem 1. *Let $\{k, m\} \in E$ with $l_{km} > 2UB$. Then, any point on this edge, except the endpoints, can be excluded from J.*

Proof. Since $l_{km} > 2$UB, either $d_{kx} >$UB or $d_{xm} >$UB for any point x on edge $\{k, m\}$. Suppose to the contrary that we place a facility at point x such that $0 < d_{xk} <$UB. In this case, $d_{xm} >$UB meaning that x cannot serve m within UB. Then, we can remove the facility from x and put it on k since we can still serve the same set of nodes and possibly some additional nodes. Therefore, any intersection point on this edge can be excluded from J, but J should contain the endpoints since they are in N. □

Theorem 2. *Let $\{k, m\} \in E$ and $i, j \in N : i \neq j$. If $d_{ik} + l_{km} + d_{mj} > 2UB$ and $d_{jk} + l_{km} + d_{mi} > 2UB$, then, we do not need to look for an intersection point associated with node pair i, j on edge $\{k, m\}$.*

Proof. Let x (if exists) be the unique point on edge $\{k, m\}$ that satisfies $d_{ik}+d_{kx} = d_{xm} + d_{mj}$ and y (if exists) be the unique point on edge $\{k, m\}$ that satisfies $d_{jk} + d_{ky} = d_{ym} + d_{mi}$. Obviously, neither x nor y can serve i and j within UB. Then, either i and j are served by different points on edge $\{k, m\}$, which can be the endpoints k, m, or the intersection points associated with some other node

pair on edge $\{k, m\}$, or by some other points on another edge. Therefore, we do not need to consider the intersection point associated with node pair i, j on edge $\{k, m\}$. □

Note that a point on some edge can suffice as an intersection point for different pairs of nodes, that is, some point x on edge $\{k, m\}$ can be an intersection point associated with both node pairs $i, j \in N$ and $r, s \in N$. Thus, any point x excluded from J for pair $i, j \in N$, can be included for another node pair $r, s \in N$.

Theorem 3. *If $d_{ik} + l_{km} + d_{jm} < LB$ or $d_{im} + l_{km} + d_{jk} < LB$, we do not need to add any intersection point to P for node pair $i, j \in N$ on edge $\{k, m\}$.*

Proof. If $d_{ik} + l_{km} + d_{jm} <$LB, then for any point θ on edge $\{k, m\}$, $d_{ik} + d_{k\theta} <$LB and $d_{\theta m} + d_{jm} <$LB, which implies that θ can serve both i and j within the lower bound. Similarly, if $d_{im} + l_{km} + d_{kj} <$LB, then $d_{\theta k} + d_{kj} <$LB and $d_{im} + d_{m\theta} <$LB, which implies that θ can serve both i and j within the lower bound. Since i, j can be served by any point on edge $\{k, m\}$, it can be served by either the intersection point of some other node pair on $\{k, m\}$ or the endpoints k, m. Since the endpoints k, m will be included in J, we guarantee that there is a point in J that can serve both i and j on this edge within the lower bound . □

We give the details of our method for generating the intersection points in Algorithm 2. See also Figure 5.1 for an illustration of intersection points on some edge $\{k, m\}$ for some node pair i, j.

In the next section, we present improved lower and upper bounds to use in our computational experiments.

5.2 Improved Lower and Upper Bounds

In order to restrict the set of distinct radius values we devise new lower and upper bounds by using the relationship between the optimal values of the absolute

Algorithm 2 Generation of Intersection Points

```
    E = {e_1, ..., e_Ω}, P ← ∅, R ← ∅, LB,UB.
 1: for s = 1 to Ω do
 2:     Let k, m be the endpoints of e_s.
 3:     if l_km ≤ 2UB and l_km ≤ d_km then
 4:         for i = 1 to n do
 5:             for j = i + 1 to n do
 6:                 d_1 = d_ik + d_jm + l_km, d_2 = d_im + d_jk + l_km
 7:                 if d_1 < d_2 then
 8:                     if d_1 ≤ 2UB and d_1 ≥ LB then
 9:                         if d_1/2 ≥ d_ik and d_1/2 ≥ d_jm then
10:                             Define λ on e_s and d_1/2 − d_ik units away from k.
11:                             if λ ∉ P then
12:                                 P ← P ∪ {λ}
13:                             end if
14:                         else
15:                             if d_2 ≤ 2UB and d_2 ≥ LB then
16:                                 if d_2/2 ≥ d_im and d_2/2 ≥ d_jk then
17:                                     Define λ on e_s and d_2/2 − d_jk units away from k.
18:                                     if λ ∉ P then
19:                                         P ← P ∪ {λ}
20:                                     end if
21:                                 end if
22:                             end if
23:                         end if
24:                     end if
25:                 else if d_2 < d_1 then
26:                     if d_2 ≤ 2UB and d_2 ≥ LB then
27:                         if d_2/2 ≥ d_im and d_2/2 ≥ d_jk then
28:                             Define λ on e_s and d_2/2 − d_jk units away from k.
29:                             if λ ∉ P then
30:                                 P ← P ∪ {λ}
31:                             end if
32:                         else
33:                             if d_1 ≤ 2UB and d_1 ≥ LB then
34:                                 if d_1/2 ≥ d_ik and d_1/2 ≥ d_jm then
35:                                     Define λ on e_s and d_1/2 − d_ik units away from k.
36:                                     if λ ∉ P then
37:                                         P ← P ∪ {λ}
38:                                     end if
39:                                 end if
40:                             end if
41:                         end if
42:                     end if
43:                 else if d_1 = d_2 then
44:                     if d_1/2 − d_ik = d_2/2 − d_jk then
45:                         if d_1 ≤ 2UB and d_1 ≥ LB then
46:                             if d_1/2 ≥ d_ik and d_1/2 ≥ d_jm then
47:                                 Define λ on e_s and d_1/2 − d_ik units away from k.
48:                                 if λ ∉ P then
49:                                     P ← P ∪ {λ}
50:                                 end if
51:                             end if
52:                         end if
53:                     end if
54:                 end if
55:             end for
56:         end for
57:     end if
58: end for
```

and vertex restricted p-center problems. Recall that r_V^* and r_A^* denote optimal values of the vertex restricted and the absolute p-center problems, respectively. Since any feasible solution to the vertex restricted p-center problem is feasible for also the absolute p-center problem, we have $r_V^* \geq r_A^*$. Therefore, the optimal solution of the vertex restricted p-center problem is an upper bound for the absolute p-center problem. In order to obtain a better upper bound, we make an improvement on the optimal solution that we obtained from the vertex restricted p-center problem. The improvement procedure is as follows:

We obtain an optimal solution for the vertex restricted p-center problem with centers $x_1, ..., x_p$ and assign each demand point to its closest center. For each center $x_j, j = 1, ..., p$ we can construct an assignment tree by connecting x_j to its demand points. If we find an absolute 1-center on each of these trees, we obtain a new solution for the absolute p-center problem. Since the absolute center of a tree is placed in the middle of its longest path [20], we find the longest path length T_j of tree associated with each center $x_j, j = 1, ..., p$. The objective value of the obtained solution is equal to $\max_{j=1,...,p} T_j/2$. Therefore, $\max_{j=1,...,p} T_j/2$ is an upper bound for the absolute p-center problem. This improved upper bound will be referred to as UB_A in the rest of this chapter.

Now, we show that the optimal value of the absolute p-center problem cannot be less than half of the optimal value of the vertex restricted p-center problem if the distance matrix satisfies the triangular inequality.

Theorem 4. $\frac{1}{2} r_V^* \leq r_A^*$ *if the triangular inequality is satisfied.*

Proof. Suppose that $r_A^* < (r_V^*/2)$ and $x_1, ..., x_p$ are the centers in an optimal solution for the absolute p-center problem. Let $N_1, ..., N_p$ be the sets of demand points that can be served by $x_1, ..., x_p$, respectively. In this case, for any $N_j, j = 1, ..., p$ we have $d_{ix_j} \leq r_A^*, \forall i \in N_j$. Then, by selecting an arbitrary vertex i_j from each $N_j, j = 1, ..., p$ we can construct a set $Y = \{i_1, ..., i_p\}$. Note that $d_{ki_j} \leq d_{kx_j} + d_{x_j i_j} \leq 2r_A^* < r_V^*, \forall k \in N_j, j = 1, ..., p$ and this implies that there exists a vertex restricted p-center Y with solution value strictly less than r_V^*, which is a contradiction. Thus, we can conclude that $r_V^* \leq 2r_A^*$. □

We use $r_V^*/2$ as a lower bound and call this lower bound as LB_A.

5.3 Computational Experiments

In our computational experiments, we use the uncapacitated p-median data from OR-Library [15]. The original data consists of a listing of edges and their lengths. In some of the instances, there are more than one edge with different lengths between same pair of nodes; in other words, the same edge has multiple lengths. In these situations, we keep only the last length of the corresponding edge as suggested in [15]. We calculate the shortest path distances between the vertices by using Floyd's [62] algorithm.

We obtain P by using Algorithm 2. During this algorithm we utilize UB_A and LB_A to reduce P without changing the optimal value of the problem. We define each intersection point with its distance to the endpoints of its edge (steps 10, 20, 34, and 44 of Algorithm 2). More specifically, suppose we find an intersection point x on edge $\{k, m\}$ that satisfies $d_{ik} + d_{kx} = d_{xm} + d_{mj}$. We store d_{kx} value and use it in construction of the distance matrix $D = [d_{ij}] : i \in N, j \in J$ after completing Algorithm 2. Once we have D, we construct $R = \{\rho_1, \ldots, \rho_M\}$ from the distinct entries of D, which are between UB_A and LB_A, and apply DBR2 algorithm to solve the absolute p-center problem.

Table 5.1 shows the results for solving the absolute p-center problem on 40 p-median instances. In this table, the first four columns give the characteristics of the problem instances. The numbers reported in the fourth column are the numbers of edges remaining after the elimination of repeated length values of the edges. In the fifth column, we provide the optimal value of the vertex restricted p-center problem solved with DBR2 for the same data. In the sixth column, we give the improved upper bound value calculated by the algorithm discussed in Section 5.2 and in the seventh column, we give the optimal value of the absolute p-center problem. The numbers given in Columns 8 and 9 represent the number of intersection points generated in the preprocessing phase of our method and

the number of distinct distance values between the vertices and the potential facilities, respectively. In the last two columns, "PP Time" is the sum of time spent for solving the vertex restricted p-center problem, obtaining the improved upper bound, and generating the intersection points while "Total Time" is the sum of "PP Time" and the time required to solve the absolute p-center problem after the preprocessing procedure. We observe from Table 5.1 that the largest preprocessing time is 60.68 seconds for the instance with $n = 900$, $p = 5$. The total time spent for solving the same instance is 508.70 seconds and it is the largest total time requirement among the 40 instances. The average preprocessing time and total time is 9.59 seconds and 73.61 seconds, respectively. With the method discussed in this chapter, we are able to solve problems with n up to 900, $|E|$ up to 16056, and $|J|$ up to 92501 (91701+800), each in less than 509 seconds. The average and worst case solution times reported in [10] are 1781 seconds and 5928 seconds, respectively. Although it is not straightforward to compare the solution times on different machines, our algorithm is expected to work more efficiently since the problem sizes are decreased with the utilization of the lower and upper bounds effectively.

Another result that we observe from this table is that UB_A is smaller than r_V^* in 13 instances and these improvements in the upper bounds improve the solution times for some of the instances. When the lower bound is set to zero instead of $r_V^*/2$, the solution time of pmed22 goes up to 220.5 seconds. Similarly, if instance pmed38 is solved with upper bound $r_V^* = 29$ instead of $UB_A = 28.5$, the solution time goes up to 486 seconds. Therefore, we see that even a slight decrease in the upper bound value or increase in the lower bound value may result in a considerable amount of decrease in the solution time. These observations reveal the importance of obtaining quality bounds for the absolute p-center problem.

Table 5.1: Results for solving absolute p-center problem via DBR2 on OR-Library instances

Instance	n	p	$\|E\|$	r_V^*	UB_A	r_A^*	$\|P\|$	$\|R\|$	PP Time (Sec)	Total Time (sec)
pmed1	100	5	200	127	119	115.5	12308	112	0.17	3.51
pmed2	100	10	196	98	98	91.5	7571	99	0.48	2.99
pmed3	100	10	199	93	92	86	6369	92	0.05	1.81
pmed4	100	20	199	74	73.5	63.5	2408	74	0.02	0.30
pmed5	100	33	197	48	44	36.5	983	41	0.02	0.27
pmed6	200	5	793	84	82	81.5	43794	81	1.00	22.48
pmed7	200	10	791	64	63.5	60.5	26814	64	0.44	11.88
pmed8	200	20	797	55	55	47.5	21041	56	0.33	2.83
pmed9	200	40	791	37	36.5	31	7057	37	0.16	0.94
pmed10	200	67	790	20	20	16.5	1975	21	0.06	0.33
pmed11	300	5	1789	59	59	56	47623	60	2.56	56.27
pmed12	300	10	1779	51	50.5	49.5	43041	51	2.46	34.94
pmed13	300	30	1779	36	36	32	22933	37	0.81	9.89
pmed14	300	60	1780	26	26	21.5	10548	27	0.53	2.15
pmed15	300	100	1775	18	18	13.5	4143	19	0.36	1.00
pmed16	400	5	3177	47	46	46	52603	46	4.73	24.99
pmed17	400	10	3178	39	38.5	37.5	50382	39	3.38	58.93
pmed18	400	40	3164	28	28	25.5	29068	29	2.04	21.96
pmed19	400	80	3166	18	17.5	15.5	9305	18	1.14	3.15
pmed20	400	133	3161	13	13	10.5	3888	14	0.86	2.01
pmed21	500	5	4954	40	40	38.5	67584	41	9.70	158.82
pmed22	500	10	4948	38	38	36	69609	39	7.71	175.07
pmed23	500	50	4949	22	22	20	29836	23	3.82	19.08
pmed24	500	100	4957	15	15	12.5	11333	16	2.65	6.50
pmed25	500	167	4941	11	11	8.5	5036	12	1.72	3.76
pmed26	600	5	7139	38	37.5	36.5	74472	38	17.41	202.74
pmed27	600	10	7138	32	32	30.5	67352	33	12.31	135.66
pmed28	600	60	7133	18	18	16	30293	19	6.58	26.03
pmed29	600	120	7126	13	13	11	14372	14	4.96	12.14
pmed30	600	200	7122	9	9	7.5	4842	10	3.28	6.96
pmed31	700	5	9713	30	30	29.5	77514	31	23.31	130.34
pmed32	700	10	9694	29	29	28	78410	30	19.94	207.02
pmed33	700	70	9703	15	15	14	30671	16	10.39	125.41
pmed34	700	140	9678	11	11	9.5	14585	12	7.58	17.06
pmed35	800	5	12670	30	30	29	81867	31	47.60	248.18
pmed36	800	10	12673	27	27	27	91701	28	32.82	172.03
pmed37	800	80	12674	15	15	13.5	41467	16	19.10	49.54
pmed38	900	5	16053	29	28.5	28	90859	29	60.68	394.30
pmed39	900	10	16056	23	23	23	83817	24	43.20	508.70
pmed40	900	90	16046	13	13	12	38793	14	27.30	82.28
Average:									9.59	73.61

Chapter 6

Single Allocation Capacitated p-Center Problem

In this chapter, we focus on the generalized capacitated p-center problem with single allocation. In this problem, each demand point has a positive demand that is not necessarily the same amount for each demand point, and each facility has a service capacity, which can also be different for distinct facilities and the objective is to locate p facilities on a given network so that the maximum distance between a demand point and the facility, to which it is assigned, is minimized and the capacity restrictions are obeyed. We propose two new mathematical formulations, and one of these formulations can be tightened as done in Chapter 4 for P3. In addition to the mathematical models, we develop an exact algorithm that we call as 'successive p-center-allocation' algorithm. The main idea of this algorithm is to solve the problem by dividing it into smaller problems. We conduct computational experiments to see the performance of our algorithm by using three different data sets which contain problems with loose as well as tight and identical as well as non-identical capacities.

In the next section we give the mathematical formulations that we propose for the single allocation capacitated p-center problem. We provide a tightened constraint and some valid inequalities for one of our formulations. In Section

6.2, we present the successive p-center-allocation algorithm and in Section 6.3, we provide the computational results regarding our algorithm. Finally, in Section 6.4 we make some conclusions.

6.1 Proposed Formulations

For simplicity, we assume that each demand node is a potential facility, i.e. $J = N$, but our methods are adaptable to the cases where the sets of demand points and facilities are not necessarily identical. As in the p-center problem, exactly one of the radius values in R determines the optimal value of the capacitated p-center problem. Recall that $y_i = 1$ if there is a facility at node $i \in N$ and 0 otherwise and $z_k = 1$ if ρ_k is equal to the optimal value and 0 otherwise for $k \in T$. We define $x_{ij} = 1$ if demand node $i \in N$ is allocated to facility $j \in N$ and 0 otherwise.

Our first formulation is as follows:

$$
\begin{aligned}
\text{(CP1)} \qquad \min\ & (4.12) && && \\
\text{s.t.}\ & \sum_{j \in N} x_{ij} \geq 1, && \forall i \in N, && (6.1)\\
& \sum_{i \in N} h_i x_{ij} \leq K_j y_j, && \forall j \in N, && (6.2)\\
& x_{ij} \leq y_j, && \forall i, j \in N, && (6.3)\\
& x_{ij} \leq 1 - \sum_{k \in T : d_{ij} > \rho_k} z_k, && \forall i, j \in N, && (6.4)\\
& \sum_{j \in N} y_j \leq p, && && (6.5)\\
& x_{ij} \in \{0, 1\}, && \forall i, j \in N, && (6.6)\\
& y_j \in \{0, 1\}, && \forall j \in N, && (6.7)\\
& (4.14), (4.15). && &&
\end{aligned}
$$

Constraint (6.1) assures that each demand point takes service from a facility. By constraint (4.14) only one of the distinct distance values is selected and this value

becomes the optimal objective value by objective function (4.12). Constraints (6.2) ensure that the total demand assigned to any facility does not exceed its capacity. Constraints (6.3) force that there is a facility at node $j \in N$ if any node is assigned to node j. Constraint (6.4) eliminates the assignments with distance values greater than the selected optimal distance value and constraint (6.5) restricts the number of facilities to p.

Our second formulation is as follows:

$$
\begin{aligned}
\text{(CP2)} \quad \min \ & (4.12) \\
\text{s.t.} \ & \sum_{j \in N_{\rho_k}(i)} x_{ij} \geq z_k, && \forall i \in N, k \in T, \quad (6.8) \\
& \sum_{i \in N_{\rho_k}(j)} h_i x_{ij} \leq K_j^{\rho_k} y_j, && \forall j \in N, k \in T, \quad (6.9) \\
& (4.14), (4.15), (6.3), (6.5), (6.6), \text{ and } (6.7)
\end{aligned}
$$

Constraints (6.8) ensure that each demand node is assigned to a facility within the selected radius value. One may question the validity of constraints (6.9) in this model, we clear out this question as follows: since $K_j^{\rho_k} = \min\{K_j, \sum_{i \in N_{\rho_k}(j)} h_i\}$, constraints (6.9) can be expressed as $\sum_{i \in N_{\rho_k}(j)} h_i x_{ij} \leq \min\{K_j, \sum_{i \in N_{\rho_k}(j)} h_i\} y_j, \forall j \in N, k \in T$. Then, we must have $\sum_{i \in N_{\rho_k}(j)} h_i x_{ij} \leq K_j y_j$ and $\sum_{i \in N_{\rho_k}(j)} h_i x_{ij} \leq \sum_{i \in N_{\rho_k}(j)} h_i y_j, \forall j \in N, k \in T$. The first expression is valid since $\sum_{i \in N_{\rho_k}(j)} h_i x_{ij} \leq \sum_{i \in N} h_i x_{ij} \leq K_j y_j, \forall j \in N, k \in T$ and the latter one is valid due to constraints (6.3).

In this model, we can use constraints (6.2) of CP1 instead of constraints (6.9), but we aim to have stronger inequalities by utilizing the effective capacities. For $k = M$, constraints (6.9) are equivalent to $\sum_{i \in N} h_i x_{ij} \leq (\min\{K_j, \sum_{i \in N} h_i\}) y_j, \forall j \in N$ and these constraints are stronger than constraints (6.2) if $\sum_{i \in N} h_i < K_j$. Moreover, if we solve CP2 for a single radius value r, we obtain much stronger constraints $\sum_{i \in N} h_i x_{ij} \leq K_j^r y_j, \forall j \in N$ by using the effective capacities. Therefore, we prefer presenting CP2 with constraints (6.9).

Constraints (6.8) in CP2 can be tightened as done in Chapter 4 and a new mathematical formulation can be obtained by replacing constraints (6.8) with the

tightened ones given below:

$$\sum_{j \in N_{\rho_k}(i)} x_{ij} \geq \sum_{l=1}^{k} z_l, \qquad \forall j \in N, k \in T. \tag{6.10}$$

The new formulation that we refer to as CP2T can be summarized as follows:

$$\begin{aligned} &\min \ (4.12) \\ &\text{s.t. } (4.14), (4.15), (6.3), (6.5), (6.6), (6.7), (6.9), \text{ and } (6.10). \end{aligned}$$

We solve our models on the capacitated p-median data from OR-Library [15]. We observe that CP2 performs better in terms of the average solution times and the LP relaxation bounds of CP2T is larger.

By replacing constraints (6.6) with $x_{ij} \geq 0, \forall i, j \in N$ in CP1, CP2, or CP2T, we can obtain mathematical formulations for solving the multiple allocation capacitated p-center problem that we focus on in Chapter 7. Note that the mathematical models in [57] and [58] cannot be adapted to the multiple allocation p-center problem with the relaxation of x variables. Thus, our formulations have an advantage in that sense.

The inequality $\sum_{i \in N} K_i y_i \geq \sum_{j \in N} h_j$ is a valid inequality for our models. We can replace this inequality with the following set of stronger inequalities: $\sum_{i \in N} K_i^{\rho_k} y_i \geq \sum_{j \in N} h_j z_k, \forall k \in T$.

In the capacitated p-center problems, the following issue needs attention. In the general case of the problem, we assume that a facility which is also a demand point can be assigned to a different facility. Because this allocation might provide a better solution in terms of the optimal value and the restriction of allocating each facility to itself might result in sub-optimal solutions. However, in some applications of the problem, self-assignment might be compulsory for the facilities; in such cases the addition of the constraint $x_{ii} = y_i, i \in N$ to our models handles this requirement.

In the next section, we propose a new algorithm which is based on decomposing the problem into two subproblems and solving them sequentially and repeatedly as needed.

6.2 Successive p-Center-Allocation Algorithm

This algorithm can be considered as a method that decomposes the capacitated p-center problem into two subproblems and solves the subproblems iteratively to obtain the optimal value of the original problem. Since this method can be applied for both allocation strategies, we will refer to both problems when we say capacitated p-center problem in this section unless it is stated explicitly. The main method of the algorithm consists of two levels. In the first level, the uncapacitated p-center problem is solved optimally and in the second level the feasibility of this optimal solution in terms of viable allocations for the capacitated p-center problem is checked. If the capacitated p-center problem is feasible, then the current solution is an optimal solution for the capacitated p-center problem as well; otherwise, the current solution is removed from the feasible region and the uncapacitated p-center problem is solved for the updated feasible region. The procedure is repeated until either a feasible solution is found for the capacitated p-center problem or the uncapacitated p-center problem becomes infeasible implying that the capacitated p-center problem is infeasible as well. Several versions for the implementation of this algorithm can be discussed, but here we provide only one version, of which we give the details below.

For an arbitrary $k \in T$ with $\rho_k = r$, we solve the following model.

$$
\begin{aligned}
(UP_k) \qquad & \min \ 0 \\
& \text{s.t.} \sum_{j \in N_r(i)} y_j \geq 1 && \forall i \in N, && (6.11) \\
& \sum_{j \in N} y_j = p && && (6.12) \\
& y_j \in \{0,1\} && \forall j \in N
\end{aligned}
$$

Decision variables and parameters in UP_k are defined as in CP1 and CP2. If UP_k is infeasible, then we need to select a larger $v \in T : v > k$; otherwise, solving UP_k provides a feasible solution $\overline{y}$ and we check if the following allocation problem is feasible for $\overline{y}$ and r.

$$
\begin{aligned}
(SAP_k) \qquad & \min\ 0 \\
& \text{s.t.} \sum_{j \in N_r(i)} x_{ij} \geq 1, && \forall i \in N, && (6.13) \\
& \sum_{i \in N_r(j)} h_i x_{ij} \leq K_j^r \overline{y_j}, && \forall j \in N, && (6.14) \\
& x_{ij} \in \{0,1\}, && \forall i,j \in N.
\end{aligned}
$$

If SAP_k has a feasible solution $\overline{x}$, then it means that there exists a feasible solution $(\overline{x}, \overline{y})$ for CP2 with objective value r; otherwise, we solve UP_k for r by excluding $\overline{y}$ from its feasible region and repeat the procedure.

In our computations we utilize the following valid inequalities when we solve UP_k.

$$
h_i(1 - y_i) \leq \sum_{j \in N_r(i) \setminus \{i\}} K_j^r y_j \qquad \forall i \in N \qquad (6.15)
$$

$$
\sum_{j \in N} K_j^r y_j \geq \sum_{i \in N} h_i. \qquad (6.16)
$$

Inequality (6.15) is redundant when $y_i = 1$ for $i \in N$ and if $y_i = 0$, at least one node in $N_r(i)$ is forced to be selected as a center. Inequality (6.16) states that the total effective capacity of the selected centers has to be greater than or equal to the total demand.

The computational experiments conducted reveal that this algorithm is able to solve most of the problems in matters of seconds; however, in some of the problems, UP_k might have a large number of alternative solutions providing input parameters that make SAP_k infeasible. In such cases, cutting these solutions takes excessive amount of time. When we solve the problems with larger range of demand values and tighter capacities, we naturally expect to eliminate more feasible solutions of UP_k since finding a feasible allocation that obey the capacities becomes more difficult. As the computational experiments support this expectation, we consider the following modification for the successive p-center-allocation algorithm. After cutting a certain number of feasible solutions of UP_k, we solve CP2 for $R = \{\rho_k\}$ (we call this problem as $CP2_k$) and decide if there exists a

feasible solution for the capacitated p-center problem for the current radius value ρ_k. Then we select a larger radius value if $CP2_k$ is infeasible or a smaller radius value if it is feasible. The details of our successive p-center-allocation algorithm with the modifications mentioned above are provided in Algorithm 3.

Algorithm 3 Successive p-center-allocation Algorithm(BL)

$\rho_1 < \rho_2 < \ldots < \rho_M$, min $\leftarrow 1$, max $\leftarrow M$, $stop \leftarrow false$, $OptVal \leftarrow \infty$.
while max $-$ min ≥ 1 **do**
  $mid \leftarrow \lfloor (\min + \max)/2 \rfloor$,
  Create an empty cut set Q.
  while $stop = false$ **do**
    if $|Q| \leq MaxCutNumber$ **then**
      Solve UP_{mid} with cuts in Q.
      if UP_{mid} has a feasible solution $\overline{y}$ **then**
        Solve SAP_{mid} for $\overline{y}$.
        if SAP_{mid} is feasible **then**
          $OptVal = \rho_{mid}$,
          max $\leftarrow mid$,
          $stop \leftarrow true$.
        else
          $Q \leftarrow Q \cup (\sum_{j \in N: \overline{y}_j = 1} y_j \leq p - 1)$.
        end if
      else
        min $\leftarrow mid + 1$,
        $stop \leftarrow true$.
      end if
    else
      Solve $CP2_{mid}$ with cuts in Q.
      if $CP2_{mid}$ is feasible **then**
        max $\leftarrow mid$,
        $OptVal = \rho_{mid}$.
      else
        min $\leftarrow mid + 1$,
      end if
    end if
  end while
end while

6.3 Computational Experiments

We conduct experiments on three different data sets. The first data set is the capacitated p-median data from OR-Library [15] and it consists of 10 instances with $n = 50, p = 5$ and 10 instances with $n = 100, p = 10$. In this data, the demand, capacity, and the location coordinates of each node is provided. The capacity is identical for all nodes. From the coordinate locations, we calculate the Euclidean distance between each node pair and round it down to the nearest integer. Although the triangular inequality is ruined by rounding, we remain safe since we do not require this property in our methods. In the rest of this section, we call this data as D1. The second data set is taken from [66] and it is a real data corresponding to the central area of São José dos Campos city. In this data, n varies from 100 to 402 and p varies from 10 to 40. We refer to this data as D2. In D2, the capacities of the facilities are identical in each single problem instance. We create a third data set, D3, by using pmed21, pmed26, pmed31, pmed35, and pmed38 networks of the uncapacitated p-median data set from OR-Library [15]. In D3, we construct three instances for each network with $p = 5, 10$, and $n/10$. We generate demand values uniformly between 1 and 500 and capacity values uniformly between 1 and $2\lceil \sum_{j \in N} h_j \frac{10}{8p} \rceil$.

When we solve the capacitated p-center problem with any of the methods we propose, we utilize the optimal value of the uncapacitated p-center problem as the lower bound for the optimal value. To obtain an upper bound value for the single allocation capacitated p-center problem, we define two methods; one of them is for the problems with identical capacities and the other one works for the ones with non-identical capacities.

- General capacities: Let F_{max} be the set of p facilities with largest capacities and $F(i) = \{j \in F_{max} : h_i \leq K_j\}$. Then, $\max_{j \in N} \max_{j \in F(i)} d_{ij}$ is an upper bound for the single allocation capacitated p-center problem.

- Equal capacities: Let $T_i = \max_{j \in N} d_{ij}, i \in N$ and $T_{i_1} \leq T_{i_2} \leq \ldots \leq T_{i_n}$. Then, $T_{i_{n-p+1}}$ is an upper bound for the single allocation capacitated p-center problem. This upper bound can be improved by comparing T_i values

with the second largest distance values to $i \in N$.

The times spent to obtain these lower and upper bounds are included in the solution times reported in the tables throughout this chapter. We compare the computational performance of our algorithm with the binary search algorithm given in Algorithm 4. In this binary search algorithm, we solve our formulation $CP2_k$.

Algorithm 4 Binary Search Algorithm(BS)

$\rho_1 < \rho_2 < \ldots < \rho_M$, $\min \leftarrow 1$, $\max \leftarrow M$, $OptVal \leftarrow \infty$.
while $\max - \min \geq 1$ **do**
  $mid \leftarrow \lfloor (\min + \max)/2 \rfloor$,
  Solve $CP2_{mid}$.
  if $CP2_{mid}$ is feasible **then**
    $OptVal = \rho_{mid}$,
    $\max \leftarrow mid$.
  else
    $\min \leftarrow mid + 1$.
  end if
end while

Tables 6.1-6.3 give the results for solving the single allocation capacitated p-center problem by using the binary search and the successive p-center-allocation algorithms on data sets D1, D2, and D3, respectively. In these tables, the columns under n, p, and Opt gives the numbers of nodes, number of facilities to be placed, and the optimal values of the problems, respectively. The columns under LB and UB show the lower and upper bound values introduced to the algorithms while solving the problems and the last two columns give the total solution times including the times needed to obtain the lower and upper bounds for the two algorithms.

From Table 6.1, we observe that in terms of the solution times, the two algorithms are not much different from each other, but binary search performs slightly better than the successive p-center-allocation algorithm not only on the average but also on the worst case.

Table 6.2 gives the results for solving the single allocation capacitated p-center

Table 6.1: Solution times (seconds) of binary search and successive p-center-allocation algorithms on D1 instances for the single allocation capacitated p-center problem

n	p	LB	UB	Opt	Binary Search Times	Successive p-center-allocation Times
50	5	29	82	29	0.95	0.76
50	5	31	77	33	0.76	0.62
50	5	26	72	26	0.64	0.36
50	5	31	70	32	0.92	1.09
50	5	27	66	29	0.64	1.02
50	5	28	76	31	1.28	4.02
50	5	30	70	30	1.04	0.75
50	5	29	81	31	0.80	1.48
50	5	27	70	28	0.87	1.78
50	5	29	76	32	0.88	0.65
100	10	19	69	19	3.94	3.94
100	10	19	75	20	3.48	6.33
100	10	19	80	20	5.08	3.39
100	10	20	76	20	5.10	2.15
100	10	20	79	21	4.96	3.64
100	10	19	70	20	6.29	8.12
100	10	20	74	22	21.57	31.35
100	10	19	76	21	6.88	6.00
100	10	20	75	21	4.35	13.73
100	10	18	84	21	8.32	19.52
				Avg.:	3.94	5.54

Table 6.2: Solution times (seconds) of binary search and successive p-center-allocation algorithms on D2 instances for the single allocation capacitated p-center problem

n	p	LB	UB	Opt	Binary Search Times	Successive p-center-allocation Times
100	10	316	1488	364	18.02	54.46
100	15	301	1631	304	18.69	30.66
300	25	276	1995	278	113.97	69.41
300	30	245	2034	253	78.13	59.62
402	30	277	2150	284	465.32	309.85
402	40	238	2214	239	373.34	350.63
				Avg.:	177.91	145.77

problem on D2 instances by using the successive p-center-allocation algorithm and the binary search algorithm. In this table, the solution times of the successive p-center-allocation algorithm are smaller than the solution times of the binary search algorithm for all instances except the first instance. Another observation on this table is that the average and the worst solution times of the successive p-center-allocation algorithm are better than those of the binary search algorithm on this data. The worst solution time of the successive p-center-allocation algorithm is 350.63 seconds while it is 465.32 seconds for the binary search algorithm. The average solution time of the binary search algorithm is 177.91 seconds and it is 145.77 seconds for the successive p-center-allocation algorithm.

Table 6.3 presents the results that we obtain from solving the single allocation capacitated p-center problem by the binary search and the successive p-center-allocation algorithms on D3 instances. In this table, we observe that all problems, except for one, are solved much faster by the successive p-center-allocation algorithm. Only the problem with $n = 600$ and $p = 60$ is solved around three times faster by the binary search algorithm. For two of the instances in this table, no solution can be found within 10 hours by using the binary search algorithm (so, the algorithm is interrupted after 10 hours); the average and the worst solution times of the remaining instances are 9934.03 seconds and 38376.68 seconds, respectively. The average and the worst solution times of all instances are 327.82 seconds and 3314.96 seconds, respectively, for the successive p-center-allocation

Table 6.3: Solution times (seconds) of binary search and successive p-center-allocation algorithms on D3 instances for the single allocation capacitated p-center problem

n	p	LB	Opt	Binary Search Times	Successive p-center-allocation Times
500	5	40	40	8979.08	9.24
500	10	34	34	2253.20	7.13
500	50	22	22	209.67	73.45
600	5	38	38	14257.20	19.25
600	10	32	32	9891.35	11.04
600	60	19	19	380.14	1183.39
700	5	30	30	9692.34	11.63
700	10	27	27	19939.28	9.89
700	70	15	19	472.21	77.22
800	5	30	30	38376.68	22.19
800	10	26	26	11152.76	19.78
800	80	14	21	2922.31	115.74
900	5	29	29	NA	19.60
900	10	24	24	NA	22.84
900	90	12	24	10616.13	3314.96
			Avg.:	9934.03	327.82

algorithm. Another observation that we obtain from this table is that, the binary search algorithm solves the instances with $p = n/10$ faster while the successive p-center-allocation algorithm solves problems with $p = 5$ or $p = 10$ faster for the same n values with the same network. In this table, the upper bound values are equal to the largest distance value; therefore, we do not give them explicitly.

6.4 Conclusion

In this chapter, we presented new mathematical formulations and a successive p-center-allocation algorithm to solve the single allocation capacitated p-center problem. We conducted large scale experiments on three different data sets; two of them were available in the literature for solving the capacitated p-median data and used for solving the capacitated p-center problem in previous studies.

We created the other one by using the uncapacitated p-median data from OR-Library [15]. We compared the performance of our successive p-center-allocation algorithm with a binary search algorithm over distinct radius values that solves one of our mathematical models for the selected fixed radius value at each step. We observed that the successive p-center-allocation algorithm performs better in terms of the solution times especially in larger problems when compared to the binary search algorithm. We were able to solve problems with up to 900 nodes by using our successive p-center-allocation algorithm.

The methods we proposed in this chapter are readily adaptable to the multiple allocation capacitated p-center problem. We leave the details of this adaptation to Chapter 7.

Chapter 7

A Branch and Cut Algorithm for Solving the Multiple Allocation Capacitated p-Center Problem

The multiple allocation capacitated p-center problem did not receive any attention in the literature. When we consider the school districting problem or medical units location problem, allocating the demand of a district to distinct facilities is meaningful and this type of allocation provides solutions as good as the ones obtained from the single allocation version in terms of the worst case service level. Therefore, it is worth to study the multiple allocation capacitated p-center problem. The mathematical models and the successive p-center-allocation algorithm that we propose for the single allocation capacitated p-center problem solve the multiple allocation capacitated p-center problem when the binary restriction of the x variables in these methods is removed. By changing x variables with $\omega : \omega_{ij} = h_i x_{ij}, \forall i, j \in N$ variables, we can obtain three additional formulations for the multiple allocation capacitated p-center problem. In the next section, we explicitly present a formulation called MCP2, which is the relaxed version of CP2, and another formulation called MCPW2, which is obtained from MCP2 by changing x variables with ω variables. Other than these methods, we propose a new branch and cut algorithm for the multiple allocation capacitated p center

problem. We give the details of our branch and cut algorithm in Section 7.2 and the experimental results that we obtained by using our branch and cut algorithm in Section 7.3.

7.1 Proposed Formulations

The parameters that are used in this section and the decision variable y is as defined in Chapter 6. Let x_{ij} be the fraction of demand of node $i \in N$ satisfied by facility $j \in N$.

$$\begin{aligned}
\text{(MCP2)} \quad & \min \ (4.12) \\
& \text{s.t.} \ (4.14), (4.15), (6.3), (6.5), (6.6), (6.7), (6.8), (6.9), \\
& \quad x_{ij} \geq 0, \forall i, j \in N. \qquad (7.1)
\end{aligned}$$

If we place ω_{ij} instead of $h_i x_{ij}$ in MCP2, we obtain the following formulation:

$$\begin{aligned}
\text{(MCPW2)} \quad & \min \ (4.12) \\
& \text{s.t.} \sum_{j \in N_{\rho_k}(i)} \omega_{ij} \geq h_i z_k, \qquad \forall i \in N, k \in T, \qquad (7.2) \\
& \quad \sum_{i \in N_{\rho_k}(j)} \omega_{ij} \leq K_j^{\rho_k} y_j, \qquad \forall j \in N, k \in T, \qquad (7.3) \\
& \quad \omega_{ij} \leq h_i y_j, \qquad \forall i, j \in N, \qquad (7.4) \\
& \quad \omega_{ij} \geq 0, \qquad \forall i, j \in N, \qquad (7.5) \\
& \quad (4.14), (4.15), (6.5), \text{ and } (6.7).
\end{aligned}$$

We can make this type of change in other models that are relaxations of CP1 and CP2T as well. The computational experiments we conducted on problems with small sizes revealed that among six models, MCP2 provides the best performance in terms of the solutions times.

7.2 A Branch and Cut Algorithm

When we consider the mathematical formulation MCP2 for a fixed radius value r, the model turns into a feasibility problem. Then, we can use $\sum_{j\in N} y_j$ or any constant (say 0) as the objective function. Below we give an appropriate formulation of this problem. As pointed out in Section 6.1, we disregard constraints (6.3) of CP2 without effecting the feasible region of the problem.

$$\text{(FMCP)} \qquad \min\ 0 \tag{7.6}$$

$$\text{s.t.} \sum_{j\in N_r(i)} x_{ij} \geq 1, \qquad \forall i \in N \tag{7.7}$$

$$\sum_{i\in N_r(j)} h_i x_{ij} \leq K_j^r y_j \qquad \forall j \in N \tag{7.8}$$

$$(6.5), (6.7), \text{ and } (7.1).$$

By projecting out the nonnegative x variables in FMCP, we obtain a new formulation with an exponential number of constraints and we develop a branch and cut algorithm to solve this new formulation by adding the constraints iteratively as they are needed. For the projection, we consider the locations of the facilities as fixed and take the dual. Let $\pi_i, i \in N$ be the dual variables associated with constraints (7.7) and $u_j, j \in N$ be the dual variables associated with constraints (7.8), then the dual problem can be expressed as follows:

$$\text{(DP)} \qquad \max \sum_{i\in N} \pi_i - \sum_{j\in N} K_j^r y_j u_j \tag{7.9}$$

$$\text{s.t. } \pi_i - h_i u_j \leq 0, \qquad \forall i, j \in N : d_{ij} \leq r \tag{7.10}$$

$$\pi_i \geq 0, \qquad \forall i \in N \tag{7.11}$$

$$u_j \geq 0, \qquad \forall j \in N. \tag{7.12}$$

Then, we analyze the extreme rays of the feasible region of the dual problem. Let us denote the feasible region of DP with W and define $N_r(S) = \bigcup_{i\in S} N_r(i)$ for any set $S \subseteq N$. The following proposition reveals the characterization of the extreme rays of W.

Proposition 3. *Let $(\pi, u) \neq 0$ be a solution to W, then it is an extreme ray of W if and only if it satisfies the following conditions:*

1. $u_i \in \{0,1\}, \forall i \in N$ *and* $\pi_i = h_i$ *or 0,* $\forall i \in N$.

2. *Let* $S = \{i \in N : u_i > 0\}$ *and* $S^* = \{i \in N : \pi_i > 0\}$. *Either* $|S| = 1$ *or* $\forall i \in S\ N_r(i) \cap S^* \neq \emptyset$.

3. *For any* $S_1^* \subset S^*$, $N_r(S_1^*) \cap N_r(S^* \setminus S_1^*) \neq \emptyset$.

Proof. (*Sufficiency:*) Suppose $(\pi, u) \in W$ is not an extreme ray of W. Then, there exist two rays $(\pi, u)^1$ and $(\pi, u)^2$ that satisfies

(i) $(\pi, u)^1, (\pi, u)^2 \in W$;

(ii) neither $(\pi, u)^1$ nor $(\pi, u)^2$ is equivalent to (π, u); and

(iii) $\frac{1}{2}(\pi, u)^1 + \frac{1}{2}(\pi, u)^2 = (\pi, u)$.

Note that $(\pi, u)^1, (\pi, u)^2 \geq 0$ due to (i); thus, $(\pi_i)^1 = (\pi_i)^2 = 0$ for $i \in N : \pi_i = 0$ and $(u_j)^1 = (u_j)^2 = 0$ for $j \in N : u_j = 0$ from (iii).

Now, suppose (π, u) satisfies 1, 2, and 3. Since $(\pi, u)^1$ is not equivalent to (π, u), there exits $i \in N$ such that either $(\pi_i)^1 \neq \pi_i$ or $(u_i)^1 \neq u_i$. In order to avoid complication, let us assume that $(\pi_1)^1 \neq \pi_1$ (note that we can do this without loss of generality). Then, either $(\pi_1)^1 = \pi_1 - \epsilon$ or $(\pi_1)^1 = \pi_1 + \epsilon$ for some sufficiently small $\epsilon > 0$.

Let us consider the case where $(\pi_1)^1 = \pi_1 - \epsilon$, note that the case with $(\pi_1)^1 = \pi_1 + \epsilon$ follows similarly. From (i), we have $h_1 - \epsilon \leq h_1(u_j)^1$ and $h_1 + \epsilon \leq h_1(u_j)^2$, $\forall j \in N_r(1) \Rightarrow h_1 - \epsilon \leq h_1 \min_{j \in N_r(1)}(u_j)^1$ and $h_1 + \epsilon \leq h_1 \min_{j \in N_r(1)}(u_j)^2$. Using (iii), we have $1 + \frac{\epsilon}{h_1} \leq \min_{j \in N_r(1)}\{2 - (u_j)^1\}$ or equivalently $1 + \frac{\epsilon}{h_1} \leq 2 - \max_{j \in N_r(1)}(u_j)^1$. Since $1 - \frac{\epsilon}{h_1} \leq \min_{j \in N_r(1)}(u_j)^1$ and $1 - \frac{\epsilon}{h_1} \geq \max_{j \in N_r(1)}(u_j)^1$, $(u_j)^1 = 1 - \frac{\epsilon}{h_1}$ and $(u_j)^2 = 1 + \frac{\epsilon}{h_1}, \forall j \in N_r(1)$. Then by (i) and (iii), $(\pi_j)^1 = h_j(1 - \frac{\epsilon}{h_1})$ and $(\pi_j)^2 = h_j(1 + \frac{\epsilon}{h_1}), \forall j \in N_r(1)$.

Now, if $S^* \setminus (N_r(1) \cap S^*) = \emptyset$, then every positive entry of $(\pi, u)^1$ and $(\pi, u)^2$ is multiplied by $(1 - \frac{\epsilon}{h_1})$ and $(1 + \frac{\epsilon}{h_1})$, respectively, since $S^* \subseteq N_r(1)$ and $S \subseteq N_r(S^*)$ by 2 and this implies that (π, u) is an extreme ray of W.

If $S^* \setminus (N_r(1) \cap S^*) \neq \emptyset$, then $\exists k \in S^* \setminus (N_r(1) \cap S^*)$. Due to 3, $N_r(1) \cap N_r(S^* \setminus \{1\}) \neq \emptyset$, so, $N_r(1) \cap N_r(k) \neq \emptyset$. Then, by (i) and (iii), $(\pi_k)^1 = h_k(1 - \frac{\epsilon}{h_1})$, $(\pi_k)^2 = h_k(1 + \frac{\epsilon}{h_1})$, $(u_k)^1 = 1 - \frac{\epsilon}{h_1}$, and $(u_k)^2 = 1 + \frac{\epsilon}{h_1}$. This implies that $(\pi_i)^1 = h_i(1 - \frac{\epsilon}{h_1})$, $(\pi_i)^2 = h_i(1 + \frac{\epsilon}{h_1})$, $(u_i)^1 = 1 - \frac{\epsilon}{h_1}$, and $(u_i)^2 = 1 + \frac{\epsilon}{h_1}, \forall i \in N_r(k)$.

Now, if $S^* \setminus ((N_r(1) \cup N_r(k)) \cap S^*) = \emptyset$, then every positive entry of $(\pi, u)^1$ and $(\pi, u)^2$ is multiplied by $(1 - \frac{\epsilon}{h_1})$ and $(1 + \frac{\epsilon}{h_1})$, respectively, and this implies that (π, u) is an extreme ray of W otherwise, we consider the partition $S_1^* = \{1, k\}$ and $S^* \setminus S_1^*$ and repeat arguments. At each iteration we multiply at least one π_i and one u_i entry of $(\pi, u)^1$ and $(\pi, u)^2$ by $(1 - \frac{\epsilon}{h_1})$ and $(1 + \frac{\epsilon}{h_1})$, respectively. Finally, we end up with $S^* \setminus (\bigcup_{i \in S_1^*} N_r(i) \cap S^*) = \emptyset$ and conclude that (π, u) is an extreme ray of W.

Now, suppose $(\pi)^1 = (\pi)^2 = \pi$ but $(u_1)^1 \neq u_1$. Then, either $0 \leq (u_1)^1 < 1$ or $(u_1)^1 > 1$. If $(u_1)^1 < 1$, then, $(\pi_j)^1 \leq h_j(u_1)^1 < h_j = \pi_j$ for some $j \in S^*$ by (i), which contradicts with $(\pi)^1 = (\pi)^2 = \pi$. If $(u_1)^1 > 1$, then $(u_1)^2 < 1$ and $(\pi_j)^2 \leq h_j(u_1)^2 < h_j = \pi_j$ for some $j \in S^*$ by (i) which contradicts with $(\pi)^1 = (\pi)^2 = \pi$.

Thus, we can conclude that (π, u) is an extreme ray of W if it satisfies properties 1, 2, and 3.

(*Necessity:*)

1. Suppose that $1 = u_{i_1} = u_{i_2} = \ldots = u_{i_k} < u_{i_{k+1}} = 1 + \delta \leq \ldots \leq u_{i_n}$. In this case, there exist solutions $(\pi, u)^1, (\pi, u)^2 \in W$ such that

$$
\begin{aligned}
&(u_{i_1})^1 = \ldots = (u_{i_k})^1 = 1 - \delta/2, \\
&(u_{i_1})^2 = \ldots = (u_{i_k})^2 = 1 + \delta/2, \\
&(\pi_j)^1 = (1 - \delta/2)\pi_j, \\
&(\pi_j)^2 = (1 + \delta/2)\pi_j, && \forall j \in \bigcup_{q=1,\ldots,k} N_r(i_q), \\
&(\pi_j)^1 = (\pi_j)^2 = \pi_j, && \forall j \notin \bigcup_{q=1,\ldots,k} N_r(i_q), \\
&(u_j)^1 = (u_j)^2 = u_j, && j = i_{k+1}, \ldots, i_n.
\end{aligned}
$$

Then, we have $\frac{1}{2}(\pi,u)^1 + \frac{1}{2}(\pi,u)^2 = (\pi,u)$ implying that (π,u) is not an extreme ray of W. Therefore, we must have $u_j = 1, \forall j \in S$ and $u_j = 0, \forall j \in N \setminus S$ for some nonempty $S \subset N$.

Moreover, note that $\pi_i = 0, \forall i \in N \setminus S^*$ is required for feasibility of (π,u).

Now, suppose to the contrary that $\pi_k = h_k - \gamma$ for sufficiently small $\gamma > 0$ for some $k \in N$. Then, there exist $(\pi,u)^1, (\pi,u)^2 \in W$ such that

$$\begin{aligned}
&(\pi_k)^1 = \pi_k - \gamma/2, \\
&(\pi_k)^2 = \pi_k + \gamma/2, \\
&(\pi_j)^1 = (\pi_j)^2 = \pi_j, && j \neq k, \\
&u^1 = u^2 = u.
\end{aligned}$$

Then, $(\pi,u)^1 \neq (\pi,u), (\pi,u)^2 \neq (\pi,u)$, and $\frac{1}{2}(\pi,u)^1 + \frac{1}{2}(\pi,u)^2 = (\pi,u)$ which implies that (π,u) cannot be an extreme ray.

2. Suppose $N_r(\kappa) \cap S^* = \emptyset$ for some $\kappa \in S : |S| \geq 2$. We can construct $(\pi,u)^1, (\pi,u)^2 \in W$ such that

$$\begin{aligned}
&(u_\kappa)^1 = u_\kappa(1-\epsilon), \\
&(u_\kappa)^2 = u_\kappa(1+\epsilon), \\
&(u_i)^1 = (u_i)^2 = u_i, && \forall i \in N \setminus \{\kappa\}, \text{ and} \\
&(\pi)^1 = (\pi)^2 = \pi.
\end{aligned}$$

Since $|S| \geq$, $(\pi,u)^1 \neq (\pi,u)$ and $(\pi,u)^2 \neq (\pi,u)$. Since $\frac{1}{2}(\pi,u)^1 + \frac{1}{2}(\pi,u)^2 = (\pi,u), (\pi,u)$ cannot be an extreme ray.

3. Suppose $\exists S_1^* \subset S^*$ such that $N_r(S_1^*) \cap N_r(S^* \setminus S_1^*) = \emptyset$. Now, consider the

following setting of $(\pi, u)^1$ and $(\pi, u)^2$:

$$
\begin{aligned}
&(\pi_i)^1 = h_i(1-\epsilon)u_i, \\
&(\pi_i)^2 = h_i(1+\epsilon)u_i, \\
&(u_i)^1 = (1-\epsilon)u_i, \text{ and,} \\
&(u_i)^2 = (1+\epsilon)u_i, 0 < \epsilon < 1, && \forall i \in N_r(S_1^*), \\
&(\pi_j)^1 = (\pi_j)^2 = \pi_j, && \forall j \in N_r(S^* \setminus S_1^*), \\
&(u_j)^1 = (u_j)^2 = u_j, && \forall j \in N_r(S^* \setminus S_1^*).
\end{aligned}
$$

Obviously, $(\pi, u)^1 \neq (\pi, u)$ and $(\pi, u)^2 \neq (\pi, u)$. Since $\frac{1}{2}(\pi, u)^1 + \frac{1}{2}(\pi, u)^2 = (\pi, u), (\pi, u)$ cannot be an extreme ray.

We proved that any $(\pi, u) \in W$ is an extreme ray of W if and only if it satisfies properties 1, 2, and 3. □

By projecting out the x variables from FMCP, we obtain the following equivalent model for fixed radius value:

$$
\begin{aligned}
\text{(MP)} \quad & \min \ 0 && (7.13) \\
& \text{s.t.} \ \sum_{i \in S} K_i^r y_i \geq \sum_{i \in S^*} h_i \quad \forall S^*, S \subseteq N \text{ with properties in Proposition 3} && (7.14) \\
& \qquad (6.5) \text{ and } (6.7)
\end{aligned}
$$

In this model, constraints (7.14) ensure that the total capacity of the nodes in S is no less than the total demand of the nodes in S^*. In other words, the demand of the nodes which have no connection with outside of S has to be served by the nodes in S. In our computations, we choose minimizing $\sum_{j \in N} y_j$ in the objective function and utilize the valid inequalities (6.15) and (6.16).

$$
\begin{aligned}
& h_i(1 - y_i) \leq \sum_{j \in N_r(i) \setminus \{i\}} K_j^r y_j && \forall i \in N && (6.15) \\
& \sum_{j \in N} K_j^r y_j \geq \sum_{i \in N} h_i. && && (6.16)
\end{aligned}
$$

In our branch and cut algorithm, we solve the following strengthened LP for the separation of inequalities (7.14).

$$
\begin{aligned}
\text{(SP)} \qquad & \max \ (7.9) \\
& \text{s.t. } (7.10), (7.11), (7.12), \\
& \quad u_j \leq \sum_{i \in N_r(j)} \pi_i, && \forall j \in N && (7.15) \\
& \quad u_j \leq 1, && \forall j \in N && (7.16)
\end{aligned}
$$

In this separation model, we aim to obtain solutions that provide positive objective values. Constraints (7.15) eliminate the extreme rays with $S^* = \emptyset, |S| = 1$. Since the objective values of this sort of extreme rays are non-positive, we do not lose generality with the addition of constraints (7.15) to our separation model. Let $(\overline{\pi}, \overline{u})$ be the optimal solution obtained from SP. If the optimal value of this solution is 0, then it means that all of constraints (7.14) in the master problem are satisfied. If the optimal value is positive, then we find a violated constraint associated with S, S^* pair where $S = \{i \in N : \overline{u_i} = 1\}$ and $S^* = \{i \in N : \overline{\pi_i} h_i\}$ and add this constraint to MP.

In order to decide on the optimal value of the multiple allocation capacitated p-center problem, we need to solve the master problem for a finite series of radius values. For this purpose, we use the binary search strategy as in the successive p-center-allocation algorithm.

7.3 Computational Experiments

We conduct experiments on two different data sets. The first one is D3 of Chapter 6 and the second one, which we call as D4, is from d1291 data of TSPLIB [16]. In D4, we have 5 instances with $p = 5, 10, 100, 129$, and 500. In d1291, the location coordinates of each node is given. We calculate the Euclidean distance between each node pair and round it down to the nearest integer. As in D3, we generate demand values uniformly between 1 and 500 and capacity values uniformly between 1 and $2\lceil \sum_{j \in N} h_j \frac{10}{8p} \rceil$.

We can obtain an upper bound for the multiple allocation p-center problem as follows: We solve FMCP for any radius value by setting $y_j = 1, \forall j \in F_{max}$, where F_{max} is the set of p facilities with largest capacities, and decide if there exists a feasible solution for that radius value. By using a search strategy, such as binary search or the ones given in Chapter 4, we are able to decide on the smallest radius value that provides a feasible solution with facility set F_{max} in polynomial time. This smallest radius value is an upper bound for the multiple allocation capacitated p-center problem. We utilize this upper bound in our computations for solving the multiple allocation capacitated p-center problem. As in the single allocation case, we utilize the optimal value of the uncapacitated p-center problem as the lower bound. The times spent on obtaining the lower and upper bounds are included in the solution times reported in the tables.

In Table 7.1, we compare the performance of our successive p-center-allocation and branch and cut algorithms for solving the multiple allocation capacitated p-center problem on D3 instances. From this table, we observe that the successive p-center-allocation algorithm performs better in terms of the solution times on the instances with $p = 5$ and $p = 10$ while the branch and algorithm performs better on the instances with $p = n/10$. When we look at the average solution times of these algorithms, we see that it is 118.13 seconds for the successive p-center-allocation algorithm and 28.84 seconds for the branch and cut algorithm. Moreover, the worst solution time of the successive p-center-allocation algorithm is 1226.16 seconds while it is 65.94 seconds for the branch and cut algorithm. Therefore, we can conclude that the branch and cut algorithm performs much better both on the average and worst case on this data set.

We solve problems in D4 by using our branch and cut algorithm and give the results in Table 7.2. We are able to solve problems with $p = 5$ and 10 optimally in less than 2 minutes. However, we are not able to solve the problems with $p = 100, 129$ and 500 optimally within the 1 hour limitation. The gap between the best upper and lower bounds we obtained for these problems are greater than 100%. The average time to solve the first two instances is 99.45 seconds.

From our experiments on D4, we observe that in our branch and cut algorithm

Table 7.1: Solution times (seconds) of our algorithms on D3 instances for the multiple allocation capacitated p-center problem

n	p	LB	UB	Opt	Successive p-center-allocation Times	Branch and Cut Times
500	5	40	54	40	8.80	16.88
500	10	34	50	34	6.64	16.42
500	50	22	41	22	47.58	32.74
600	5	38	53	38	19.87	27.77
600	10	32	53	32	9.84	19.31
600	60	19	41	19	64.66	25.15
700	5	30	44	30	11.20	26.77
700	10	27	38	27	9.84	19.39
700	70	15	31	19	98.89	45.37
800	5	30	40	30	22.15	23.29
800	10	26	36	26	16.93	36.96
800	80	14	31	21	177.93	13.10
900	5	29	59	29	23.03	37.79
900	10	24	56	24	28.45	65.94
900	90	12	44	24	1226.16	25.79
				Avg.:	118.13	28.84

Table 7.2: Solution times (seconds) of our Branch and Cut Algorithm on D4 instances for the multiple allocation capacitated p-center problem

n	p	LB	UB	Opt	Time
1291	5	1007	1805	1010	82.99
1291	10	594	1487	594	115.91
1291	100	127	1390	NA	NA
1291	129	114	1510	126	NA
1291	500	50	216	NA	NA
				Avg.:	99.45

MP model is solved faster for larger radius values, but it might take large amount of time to solve MP for small radius values. If we try to solve MP with branch and cut for large radius values and $CP2_k$ instead of MP for smaller radius values, we are able to obtain the optimal value for problem with $p = 129$ in less than half an hour by limiting the branch and cut time with 2 minutes; but, we are not able to solve problem with $p = 100$ within a reasonable time limit since $CP2_k$ takes more than 3 hours to solve for some radius values.

Chapter 8

Conclusions and Future Research Directions

8.1 Preliminary Results of a Benders Decomposition Algorithm for Solving the p-Center Problem

In order to handle difficult problems more easily, several decomposition methods are available in the literature. In this section, we focus on solving the p-center problem by using the Benders Decomposition method [67]. This method can be applied to the models that have a certain structure and we are able to formulate the p-center problem in such a way. We initially give the details of our mathematical formulation and then, go into the details of our Benders Decomposition algorithm. We present our method for the vertex restricted p-center problem, but, it can easily be adapted to the absolute p-center problem. Let x_{ij} be the amount of flow going from node $i \in N$ to facility $j \in J$. Define y and z variables as in Chapter 4. Then, the following mathematical model solves the p-center problem optimally.

$$
\begin{aligned}
\text{(BP)} \quad & \min \ (4.12) \\
& \text{s.t.} \sum_{j \in N_{\rho_k}(i)} x_{ij} \geq z_k, && \forall i \in N, k \in T, && (8.1) \\
& x_{ij} \geq 0 && \forall i \in N, j \in J, && (8.2) \\
& (4.4), (4.5), (4.6), (4.14), \text{ and } (4.15).
\end{aligned}
$$

We can solve this model by decomposing into smaller problems as we discuss in Chapter 4. Therefore, we can initially focus on solving BP for a single $r \in R$. In this restrictive model, we make a simple modification by removing constraint (4.5) and minimizing $\sum_{j \in N} y_j$ in the objective. The modified restriction is as follows:

$$
\begin{aligned}
\text{(FBP)}: \quad & \min \sum_{j \in J} y_j && && (8.3) \\
& \text{s.t.} \sum_{j \in N_r(i)} x_{ij} \geq 1, && \forall i \in N, && (8.4) \\
& (4.4), (4.6), \text{ and } (8.2).
\end{aligned}
$$

When the y variables are fixed to $\overline{y}$ in this model, we obtain a linear programming problem. The dual problem of this linear programming problem can be expressed as follows. We define α to be the set of dual variables associated with constraints (8.4) and β to be the set of variables associated with constraints (4.4).

$$
\begin{aligned}
\text{(DF)}: \quad & \max \sum_{j \in J} \alpha_i - \sum_{i \in N} \sum_{j \in N_r(i)} \beta_{ij} \overline{y_j}, && && (8.5) \\
& \text{s.t. } \alpha_i - \beta_{ij} \leq 0, && \forall i \in N, j \in J : d_{ij} \leq r, && (8.6) \\
& \alpha_i \geq 0, && \forall i \in N, && (8.7) \\
& \beta_{ij} \geq 0, && \forall i \in N, j \in J : d_{ij} \leq r. && (8.8)
\end{aligned}
$$

This dual problem DF always has a feasible solution $(\alpha, \beta) = (0, 0)$ and this solution is an extreme point of DF. Let $(\alpha, \beta)^1 \neq (0, 0)$ be any feasible solution for

DF, then, $(\alpha,\beta)^2 = 2(\alpha,\beta)^1$) is also feasible for DF and since $\frac{1}{2}(0,0)+\frac{1}{2}(\alpha,\beta)^2 = (\alpha,\beta)^1$, $(\alpha,\beta)^1$ cannot be an extreme point of DF. Thus, $(0,0)$ is the unique extreme point of DF. Let $\{(\alpha,\beta)^1,\ldots,(\alpha,\beta)^K\}$ be the set of extreme rays of of DF. The Benders reformulation for FBP can be expressed as follows:

$$
\begin{aligned}
(\text{BR}): \quad & \min\ z && && (8.9)\\
& \text{s.t. } z \geq \sum_{j\in J} y_j, && && (8.10)\\
& \sum_{i\in N} \alpha_i^k - \sum_{i\in N}\sum_{j\in N_r(i)} \beta_{ij}^k y_j \leq 0, && k=1,\ldots,K, && (8.11)\\
& y_j \in \{0,1\} && \forall j \in J.
\end{aligned}
$$

Since the number of extreme rays of DF can be excessive, we develop a cutting plane algorithm to solve the Benders reformulation. A typical repetitive step of our algorithm can be described as follows: We solve DF by using the known $\overline{y}$ values. If the problem is unbounded, we solve a modified dual problem MDF, which has the same feasible region with DF but a constant objective function, say '1' and obtain an $(\alpha,\beta)^1$ solution, which is an extreme ray of DF. We solve BR by adding the constraint associated with this extreme ray and obtain a new $(y)^1$ solution. We solve DF by using $(y)^1$ values and repeat previous steps if DF is unbounded. We terminate the algorithm when DF is bounded. The last solution we obtained from BR is an optimal solution for FBP. If the optimal value of the BR algorithm is greater than p, we can conclude that there exists no feasible solution to FBP for the current radius value. Then, we choose a larger radius value to solve FBP with the Benders algorithm.

Recall that the optimal value of the p-center problem is equal to one of the distinct distance values $R = \{\rho_1,\ldots,\rho_M\}$. Therefore, we can solve the p-center problem by using the binary search algorithm given in Chapter 4 except that we solve BR for the corresponding element of R at each step instead.

We solve 40 instances of the uncapacitated p-median data from OR-Library [15] by using our Benders Decomposition algorithm and compare with the double

bound algorithm. We observe that the Benders algorithm does not perform as good as the double bound algorithm in terms of the solution times. On the other hand, we observe that the solution times of the individual problems DF, MDF, and BR of the Benders algorithm are quite small (less than 1 second). Therefore, we try to solve some of the larger problems that cannot be solved by the double bound algorithm within the time limit. For example, we know that the optimal value of the problem with $n = 2500$ and $p = 100$ is between 1050 and 1059 from Table 4.9. We try to solve FBP for $r = 1054$ by using the Benders algorithm. During this experiment, the algorithm terminates due to memory overflow after 135 iterations and the last BR solved in this experiment provides an optimal value of 64. Since each of the problems DF, MDF, and BR can be solved in less than 2 seconds, we believe that this memory overflow occurs not due to the size of the models but due to programming related issues. Therefore, Benders algorithm might be developed into a promising method for solving large problems in the future.

8.2 Contribution Summary

In this thesis, we studied the p-center problem and the capacitated p-center problem on general networks with generalized settings. We approached these problems from a modeling and algorithmic perspective. We concentrated on both absolute and vertex restricted p-center problems on weighted and unweighted networks. We investigated the capacitated p-center problem with both single and multiple allocation strategies. Our studies on the multiple allocation capacitated p-center problem are the first ones in the literature.

We developed new mathematical formulations and algorithms that solve both absolute and vertex restricted p-center problems optimally. We obtained the tightest lower bound from a semi-relaxation of our mathematical formulations. One of our formulations provided the largest LP relaxation bound among the existing formulations in the literature. In the experimentation phase, we initially focused on the vertex restricted p-center problem. For the vertex restricted

problem, we provided new upper and lower bounds that can be obtained via a polynomial algorithm in no matter of time. These bounds are within a constant multiple of the optimal value of the problem. We conducted large scale experiments on both weighted and unweighted problems and solved problems with up to 3038 nodes. We observed that the weighted problems can be solved much faster when compared to the unweighted problems. Since the symmetry of the distance matrix is lost in the weighted version of the problem, the number of non-zero coefficients decreases for a significant number of radius values. Therefore, solving weighted problems in smaller computation times was in fact an expected result. In order to solve the absolute p-center problem on the same networks, we needed to generate the intersection points. For this purpose, we developed a method for generation of the intersection points. In this method, we utilized the new lower and upper bounds that we proposed specifically for the absolute p-center problem. We were able to solve problems with up to 900 nodes and 16056 edges. The largest solution time required was less than 509 seconds in our experiments.

As in our studies for the p-center problem, we initially focused on the modeling approaches in solving the capacitated p-center problem. Then we directed our attention to developing more effective algorithms that solve the problem optimally. We proposed several new mathematical formulations and a successive p-center-allocation algorithm to solve both the single and multiple allocation capacitated p-center problems. The first set of our experiments were on the single allocation version of the problem and we were able to solve problems with up to 900 nodes. Later, we developed a branch and cut algorithm to solve the multiple allocation capacitated p-center problem and conducted experiments on relatively larger sized problems with tight and loose, identical and non-identical capacities. We were able to solve problems with up to 1291 nodes by using our branch and cut algorithm. In our experimentations, we utilized the optimal value of the (uncapacitated) p-center problem as a lower bound for the optimal value of the single and multiple allocation capacitated p-center problems. We also used distinct upper bounds for solving the single and multiple allocation versions of the problem.

8.3 Future Research Directions

One future improvement for our study on the capacitated p-center problem might be to try to solve larger problems by utilizing the reduction rules that we used for solving the p-center problem. Applying these rules improved the solution times and best bounds obtained for several problems in Chapter 4. One may also think of applying these rules for solving the absolute p-center problem. In this case, since the allocation matrix of the absolute p-center problem is larger compared to the one in the vertex restricted p-center problem, it is expected to come up with larger reduction times. However, it might be worthy if CPLEX solution times can be decreased enormously with the help of reduction process.

Another future research direction is to extend our methods for the p-center problem to the fault tolerant p-center problem. The fault tolerant p-center problem is a generalization of the p-center problem, where each demand node must have at least α centers close to it. The main motivation of this problem is to establish a back-up option in case of failure of a center in providing service. Fault tolerance can be studied in the capacitated p-center problem as well. In addition, one may wish to add some stochastic parameters to represent the fault probabilities and minimize the maximum expected service time of demand nodes.

As revealed in the literature review, there are several studies of the p-center problem on tree networks. However, the studies that focus on the capacitated p-center problem on tree networks is restricted to a unique study on the balanced p-center problem. Therefore, analyzing the structural properties of the generalized capacitated p-center problem on special networks is an open research area.

Recent natural disasters attracted the attention of the researchers for focusing on the evacuation planning problems. One of the main concerns of the evacuation process is that people get stuck at the roads due to overloads and either it causes a big chaos or evacuation cannot be achieved on time. These overloads can be avoided if the capacities of the roads can be successfully utilized in the evacuation plan. The affected population can be directed to the safe regions that will be decided by solving a p-center problem with arc capacities.

Bibliography

[1] S. Elloumi, M. Labbé, and Y. Pochet, "A new formulation and resolution method for the p-center problem," *INFORMS Journal on Computing*, vol. 16, no. 1, pp. 84–94, 2004.

[2] E. Minieka, "The m-center problem," *SIAM Review*, vol. 12, no. 1, pp. 138–139, 1970.

[3] K. Aardal, Y. Pochet, and L. A. Wolsey, "Capacitated facility location: valid inequalities and facets," *Mathematics of Operations Research*, vol. 20, no. 3, pp. 562–582, 1995.

[4] S. L. Hakimi, "Optimum locations of switching centers and the absolute centers and medians of a graph," *Operations Research*, vol. 12, no. 3, pp. 450–459, 1964.

[5] O. Kariv and S. L. Hakimi, "An algorithmic approach to network location problems. Part I: The p-centers," *SIAM Journal on Applied Mathematics*, vol. 37, no. 3, pp. 513–538, 1979.

[6] J. Hooker, R. Garfinkel, and C. Chen, "Finite dominating sets for network location problems," *Operations Research*, vol. 39, no. 1, pp. 100–118, 1991.

[7] N. Christofides and P. Viola, "The optimum location of multi-centres on a graph," *Operational Research Quarterly*, pp. 145–154, 1971.

[8] C. Toregas, R. Swain, C. ReVelle, and L. Bergman, "The location of emergency service facilities," *Operations Research*, vol. 19, no. 6, pp. 1363–1373, 1971.

[9] R. Garfinkel, A. Neebe, and M. Rao, "The m-center problem: Minimax facility location," *Management Science*, vol. 23, no. 10, pp. 1133–1142, 1977.

[10] Y. Saç, "Implementation of new and classical set covering based algorithms for solving the absolute p-center problem," Master's thesis, Bilkent University, 2011.

[11] M. S. Daskin, *Network and discrete location: Models, Algorithms, and Applications*. New York: Wiley, 1995.

[12] M. S. Daskin, "A new approach to solving the vertex p-center problem to optimality: Algorithm and computational results," *Communications of the Operations Research Society of Japan*, vol. 45, no. 9, pp. 428–436, 2000.

[13] T. Ilhan and M. Pinar, "An efficient exact algorithm for the vertex p-center problem," *Preprint.[Online]. Available: http://www. ie. bilkent. edu. tr/∼ mustafap/pubs*, 2001.

[14] A. Al-khedhairi and S. Salhi, "Enhancements to two exact algorithms for solving the vertex p-center problem," *Journal of Mathematical Modelling and Algorithms*, vol. 4, no. 2, pp. 129–147, 2005.

[15] J. E. Beasley, "OR-LIBRARY," *http://people.brunel.ac.uk/∼mastjjb/jeb/info.html*, 2012.

[16] G. Reinelt, "TSPLIB - A traveling salesman problem library," *ORSA journal on computing*, vol. 3, no. 4, pp. 376–384, 1991.

[17] J. Moreno, "A new result on the complexity of the p-center problem," tech. rep., Technical Report, Universidad Complutense, Madrid, Spain, 1986.

[18] A. Tamir, "Improved complexity bounds for center location problems on networks by using dynamic data structures," *SIAM Journal on Discrete Mathematics*, vol. 1, no. 3, pp. 377–396, 1988.

[19] A. Goldman, "Minimax location of a facility in a network," *Transportation Science*, vol. 6, no. 4, pp. 407–418, 1972.

[20] G. Y. Handler, "Minimax location of a facility in an undirected tree graph," *Transportation Science*, vol. 7, no. 3, pp. 287–293, 1973.

[21] S. Halfin, "Letter to the editor−on finding the absolute and vertex centers of a tree with distances," *Transportation Science*, vol. 8, no. 1, pp. 75–77, 1974.

[22] P. M. Dearing and R. L. Francis, "A minimax location problem on a network," *Transportation Science*, vol. 8, no. 4, pp. 333–343, 1974.

[23] S. L. Hakimi, E. F. Schmeichel, and J. Pierce, "On p-centers in networks," *Transportation Science*, vol. 12, no. 1, pp. 1–15, 1978.

[24] N. Megiddo, A. Tamir, E. Zemel, and R. Chandrasekaran, "An $O(nlog^2n)$ Algorithm for the k-th Longest Path in a Tree with Applications to Location Problems," *SIAM Journal on Computing*, vol. 10, no. 2, pp. 328–337, 1981.

[25] B. Tansel, R. Francis, T. Lowe, and M. Chen, "Duality and distance constraints for the nonlinear p-center problem and covering problem on a tree network," *Operations Research*, vol. 30, no. 4, pp. 725–744, 1982.

[26] N. Megiddo and A. Tamir, "New results on the complexity of p-centre problems," *SIAM Journal on Computing*, vol. 12, no. 4, pp. 751–758, 1983.

[27] M. Jaeger and O. Kariv, "Algorithms for finding p-centers on a weighted tree (for relatively small p)," *Networks*, vol. 15, no. 3, pp. 381–389, 1985.

[28] D. X. Shaw, "A unified limited column generation approach for facility location problems on trees," *Annals of Operations Research*, vol. 87, pp. 363–382, 1999.

[29] E. Minieka, "A polynomial time algorithm for finding the absolute center of a network," *Networks*, vol. 11, no. 4, pp. 351–355, 1981.

[30] D. Dvir and G. Y. Handler, "The absolute center of a network," *Networks*, vol. 43, no. 2, pp. 109–118, 2004.

[31] G. Y. Handler and P. B. Mirchandani, *Location on networks: theory and algorithms*, vol. 979. MIT press Cambridge, MA, 1979.

[32] D. Chen and R. Chen, "New relaxation-based algorithms for the optimal solution of the continuous and discrete p-center problems," *Computers & Operations Research*, vol. 36, no. 5, pp. 1646–1655, 2009.

[33] C. Caruso, A. Colorni, and L. Aloi, "Dominant, an algorithm for the p-center problem," *European Journal of Operational Research*, vol. 149, no. 1, pp. 53–64, 2003.

[34] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, pp. 293–306, 1985.

[35] D. S. Hochbaum and D. B. Shmoys, "A best possible heuristic for the k-center problem," *Mathematics of operations research*, vol. 10, no. 2, pp. 180–184, 1985.

[36] D. S. Hochbaum, "When are NP-hard location problems easy?," *Annals of Operations Research*, vol. 1, no. 3, pp. 201–214, 1984.

[37] W.-L. Hsu and G. L. Nemhauser, "Easy and hard bottleneck location problems," *Discrete Applied Mathematics*, vol. 1, no. 3, pp. 209–215, 1979.

[38] J. Plesník, "A heuristic for the p-center problems in graphs," *Discrete Applied Mathematics*, vol. 17, no. 3, pp. 263–268, 1987.

[39] D. B. Shmoys, "Computing near-optimal solutions to combinatorial optimization problems," *Combinatorial Optimization*, vol. 20, pp. 355–397, 1995.

[40] N. Mladenović, M. Labbé, and P. Hansen, "Solving the p-center problem with tabu search and variable neighborhood search," *Networks*, vol. 42, no. 1, pp. 48–64, 2003.

[41] S. Salhi and A. Al-Khedhairi, "Integrating heuristic information into exact methods: The case of the vertex p-centre problem," *Journal of the Operational Research Society*, vol. 61, no. 11, pp. 1619–1631, 2009.

[42] W. Pullan, "A memetic genetic algorithm for the vertex p-center problem," *Evolutionary Computation*, vol. 16, no. 3, pp. 417–436, 2008.

[43] T. Davidović, D. Ramljak, M. Šelmić, and D. Teodorović, "Bee colony optimization for the p-center problem," *Computers & Operations Research*, vol. 38, no. 10, pp. 1367–1376, 2011.

[44] J. S. Martinich, "A vertex-closing approach to the p-center problem," *Naval Research Logistics (NRL)*, vol. 35, no. 2, pp. 185–201, 1988.

[45] B. Bozkaya and B. Tansel, "A spanning tree approach to the absolute p-center problem," *Location Science*, vol. 6, no. 1, pp. 83–107, 1998.

[46] J. Mihelič and B. Robič, *Approximation algorithms for the k-center problem: an experimental evaluation*. Springer, 2003.

[47] B. Robič and J. Mihelič, "Solving the k-center problem efficiently with a dominating set algorithm," *Journal of Computing and Information Technology*, vol. 13, no. 3, pp. 225–234, 2005.

[48] B. C. Tansel, R. L. Francis, and T. J. Lowe, "State of the Art − Location on Networks: A Survey. Part I: The p-Center and p-Median Problems," *Management Science*, vol. 29, no. 4, pp. 482–497, 1983.

[49] B. C. Tansel, R. L. Francis, and T. J. Lowe, "State of the Art − Location on Networks: A Survey. Part II: Exploiting Tree Network Structure," *Management Science*, vol. 29, no. 4, pp. 498–511, 1983.

[50] T. S. Hale and C. R. Moberg, "Location science research: a review," *Annals of Operations Research*, vol. 123, no. 1-4, pp. 21–35, 2003.

[51] C. S. ReVelle and H. A. Eiselt, "Location analysis: A synthesis and survey," *European Journal of Operational Research*, vol. 165, no. 1, pp. 1–19, 2005.

[52] B. C. Tansel, "Discrete center problems," in *Foundations of location analysis* (H. A. Eiselt and V. Marianov, eds.), ch. 5, pp. 79–106, New York: Springer, 2011.

[53] J. Bar-Ilan, G. Kortsarz, and D. Peleg, "How to allocate network centers," *J. Algorithms*, vol. 15, no. 3, pp. 385–415, 1993.

[54] S. Khuller and Y. J. Sussmann, "The capacitated k-center problem," *SIAM Journal on Discrete Mathematics*, vol. 13, no. 3, pp. 403–418, 2000.

[55] M. Cygan, M. Hajiaghayi, and S. Khuller, "LP rounding for k-centers with non-uniform hard capacities," in *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pp. 273–282, IEEE, 2012.

[56] M. Jaeger and J. Goldberg, "Technical Note - A Polynomial Algorithm for the Equal Capacity p-Center Problem on Trees," *Transportation Science*, vol. 28, no. 2, pp. 167–175, 1994.

[57] M. P. Scaparra, S. Pallottino, and M. G. Scutellà, "Large-scale local search heuristics for the capacitated vertex p-center problem," *Networks*, vol. 43, no. 4, pp. 241–255, 2004.

[58] A. F. Özsoy and M. C. Pınar, "An exact algorithm for the capacitated vertex p-center problem," *Computers & Operations Research*, vol. 33, no. 5, pp. 1420–1436, 2006.

[59] M. Albareda-Sambola, J. A. Díaz, and E. Fernández, "Lagrangean duals and exact solution to the capacitated p-center problem," *European Journal of Operational Research*, vol. 201, no. 1, pp. 71–81, 2010.

[60] H. Calik and B. C. Tansel, "Double bound method for solving the p-center location problem," *Computers & Operations Research*, vol. 40, no. 12, pp. 2991–2999, 2013.

[61] G. Cornuéjols, G. L. Nemhauser, and L. A. Wolsey, "A canonical representation of simple plant location problems and its applications," *SIAM Journal on Algebraic Discrete Methods*, vol. 1, no. 3, pp. 261–272, 1980.

[62] R. W. Floyd, "Algorithm 97: Shortest path," *Communications of ACM*, vol. 5, no. 6, pp. 345–345, 1962.

[63] XINOX Software, "JCreator - Java IDE," *http://www.jcreator.com/*, 2012.

[64] IBM, "IBM ILOG CPLEX Optimization Studio," *www.ibm.com/software /products/en/ibmilogcpleoptistud/*, 2013.

[65] R. L. Francis, J. Leon F. McGinnis, and J. A. White, *Facility Layout and Location: An Analytical Approach*. Upper Saddle River, NJ 07458: PRENTICE HALL, 1992.

[66] L. A. Lorena and E. L. Senne, "A column generation approach to capacitated p-median problems," *Computers & Operations Research*, vol. 31, no. 6, pp. 863–876, 2004.

[67] J. F. Benders, "Partitioning procedures for solving mixed-variables programming problems," *Numerische mathematik*, vol. 4, no. 1, pp. 238–252, 1962.