IMPLEMENTATION OF NEW AND CLASSICAL SET COVERING BASED ALGORITHMS FOR SOLVING THE ABSOLUTE P-CENTER PROBLEM

A THESIS

SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By
Yiğit Saç
January 2011

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

_____

Prof. Dr. Barbaros Tansel (Supervisor)

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

_____

Assoc. Prof. Dr. Bahar Yetiş Kara

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

_____

Assoc. Prof. Dr. Burçin Bozkaya

Approved for the Institute of Engineering and Science

_____

Prof. Dr. Levent Onural

Director of Institute of Engineering and Science

ABSTRACT

IMPLEMENTATION OF NEW AND CLASSICAL SET COVERING BASED ALGORITHMS FOR SOLVING THE ABSOLUTE P-CENTER PROBLEM

Yiğit Saç

M.S. in Industrial Engineering

Supervisor: Prof. Dr. Barbaros Tansel

January, 2011

The p-center problem is a model of locating p facilities on a network in order to minimize the maximum coverage distance between each vertex and its closest facility. The main application areas of p-center problem are emergency service locations such as fire and police stations, hospitals and ambulance services. If the p facilities can be located anywhere on a network including vertices and interior points of edges, the resulting problem is referred to as the *absolute* p-center problem and if they are restricted to vertex locations, it is referred to as the *vertex-restricted* problem. The absolute p-center problem is considerably more complicated to solve than the vertex-restricted version. In the literature, most of the computational analysis and new algorithm developments are performed through the vertex restricted case of the p-center problem. The absolute p-center problem has received much less attention in the literature. In this thesis, our focus is on the absolute p-center problem based on an algorithm for the p-center problem proposed by Tansel (2009). Our work is the first one to solve large instances up to 900 vertices on the absolute p-center problem. The algorithm focuses on solving the p-center problem with a finite series of minimum set covering problems, but the set covering problems used in the algorithm are constructed differently compared to the ones traditionally used in the literature. The proposed algorithm is applicable for both absolute and vertex-restricted p-center problems with weighted and unweighted cases.

Keywords: p-center problem, absolute p-center problem, vertex-restricted p-center problem

ÖZET

MUTLAK P-MERKEZ PROBLEMİNİN ÇÖZÜMÜ İÇİN YENİ VE KLASİK KÜME KAPLAMA TABANLI ALGORİTMALAR VE UYGULAMASI

Yiğit Saç
Endüstri Mühendisliği Yüksek Lisans
Tez Yöneticisi: Prof. Dr. Barbaros Tansel
Ocak, 2011

P-merkez problemi, talepler ile taleplere en yakın tesisler arasındaki uzaklıkların en büyüğünü en küçükleyecek şekilde p tane tesisin yerlerinin seçimi problemidir. P-merkez probleminin temel uygulama alanları genellikle acil hizmet servislerinden oluşmaktadır. Polis karakolu, itfaiye, hastane ve ambulans servisleri genellikle p-merkez problemi esaslarına göre yerleştirilirler. Bütün bu uygulamalardaki temel öncelik insan hayatının kurtarılmasına yöneliktir. Gerçek hayata yönelik uygulamalarında p-merkez problemi çoğunlukla mutlak p-merkez problemi olarak incelenmiştir. Mutlak p-merkez probleminde yerleştirilecek olan merkezlerin mevkileri üzerinde herhangi bir kısıt bulunmamaktadır. Ancak düğüm kısıtlı p-merkez probleminde merkezler sadece şebekenin düğüm noktalarına yerleştirilebilmektedir. Mutlak p-merkez probleminde yerleştirecek olan merkezlerin mevkileri üzerinde herhangi bir kısıt bulunmaması, problemi düğüm kısıtlı şeklinden daha karmaşık hale getirmektedir. Mutlak p-merkez problemi, karmaşık yapısından dolayı gerek yeni algoritma geliştirmede gerekse sayısal analiz çalışmalarında düğüm kısıtlı p-merkez problemine göre literatürde fazla yer almamıştır. Bu tez çalışmasında Tansel (2009) tarafından önerilen p-merkez problem algoritması işlenmiş ve üzerinde sayısal analiz yapılmıştır. Önerilen ve üzerinde sayısal analiz yapılan algoritmanın uygulanabilirliği ağırlıklı, ağırlıksız mutlak ve düğüm kısıtlı p-merkez problemleri için geçerlidir.

Anahtar Kelimeler: p-merkez problemi, mutlak p-merkez problemi, tepe nokta kısıtlı p-merkez problemi

ACKNOWLEDGEMENT

Foremost, I would like to express my sincere gratitude to my supervisor, Prof. Dr. Barbaros Tansel for the continuous support of my M.S study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better supervisor and mentor for my M.S study.

Special thanks to Assoc. Prof. Dr. Bahar Yetiş Kara and Assoc. Prof. Dr. Burçin Bozkaya for accepting to read and review this thesis and for their substantial suggestions.

I would like to thank to Hatice Çalık and Emre Uzun for their great friendship during my graduate study. My sincere thanks go to my friends Ahmet Korhan Aras, Gülşah Hançerlioğulları, Ece Zeliha Demirci, Burak Paç, Esra Koca,Yahya Saleh, Fevzi Yılmaz, Ali Can Ergür and all other friends for providing me such a friendly environment to work.

I would expressly like to thank to love of my life Nazlı Avcı for her eternal love and support during my M.S study. The completion of this thesis would be impossible without her.

Last but not the least; I would like to thank to my lovely family. The constant inspiration and guidance kept me focused and motivated. I am grateful to my dad Yaşar Saç for giving me the life I ever dreamed. I can't express my gratitude for my mom Berrin Saç in words, whose unconditional love has been my greatest strength. The constant love and support of my sister Berfin Saç is sincerely acknowledged.

I would like to thank everybody who was important to the successful realization of thesis, as well as expressing my apology that I could not mention personally one by one.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION 1
CHAPTER 2: LITERATURE REVIEW 5
2.1 The Absolute 1-Center Problem 6
2.2 The Absolute p-Center Problem 8
2.3 The Vertex-Restricted p-Center Problem 10
CHAPTER 3: ALGORITHM & MATHEMATICAL FORMULATION 14
3.1 Notations of the Algorithm 15
3.2 Definitions 16
3.1.1 Distance Function of a Point on a Given Edge 16
3.1.2 Definition of Intersection Point 18
3.1.3 Definition of Radius 19
3.1.4 Definition of an Antipodal 20
3.3 Algorithm for Computing p-Center Problem 23
3.4 Illustration of the Algorithm with an Example 27
3.5 Comparison Between Proposed Method and Classical Method 45
CHATER 4: IMPLEMENTATION 53
4.1. Implementation of the Algorithm 53
CHAPTER 5: COMPUTATIONAL ANALYSIS 60
5.1. Test Problems 60
5.2. Performance Measures 61
5.3. Developing the Turkish Road Network 62
5.4. Computational Results for Absolute P-center Problem 63
5.5. Computational Comparison between Absolute and Vertex-Restricted Networks 67
5.6. Computational Comparison between Complete and Incomplete Networks 70
5.7. Investigation of LP Relaxation Bounds 74
CHAPTER 6: CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS 77
BIBLIOGRAPHY 80

LIST OF FIGURES

Figure 1: The illustration of shortest path distances 16

Figure 2: The illustration of shortest path distance functions 17

Figure 3: The illustration of an intersection point 18

Figure 4: Illustration of an intersection point-radius pair 20

Figure 5: The illustration of an antipodal 21

Figure 6: The illustration of an antipodal and corresponding distance functions 22

Figure 7: The illustration of the network 27

Figure 8: The illustration of the conditions for evaluating antipodal 29

Figure 9: The illustration of an antipodal on an edge 30

Figure 10: Illustration of newly formed edges on the network 31

Figure 11: Illustration of an intersection point by piecewise linear distance functions 33

Figure 12: The illustration of an intersection point 35

Figure 13: The trees formed by intersection point-radius pairs of the example 40

Figure 14: Illustration of the trees formed by the centers resulted from example 44

Figure 15: The location of the centers on network 45

Figure 16: Comparison between classical and proposed method 48

Figure 17: The matrix A of the proposed method 49

Figure 18: The matrix A of the classical method 49

Figure 19: Selected columns of matrix A in the example 50

Figure 20: Construction of matrix A in proposed and classical method 51

Figure 21: The comparison between proposed and classical method algorithm 52

LIST OF TABLES

Table 1: The size of the test problems 61

Table 2: Computational results for absolute p-center problem 64

Table 3: Comparison between vertex-restricted and absolute cases 68

Table 4: Computational results for incomplete network 70

Table 5: Computational results for complete network 71

Table 6: Computational comparison between planar and complete network 73

Table 7: Computational results for the LP relaxation and the IP formulation 75

Chapter 1

INTRODUCTION

The p-center problem is a model of locating p facilities on a network in order to minimize the maximum coverage distance between each vertex (demand point) and its closest facility. In p-center problems, we partition a set of demand points into exactly p subsets each associated with a center. A center is identified both by the location of its facility and by the set of demand points assigned to it. There is a given weight for assigning each demand point to each center, and we want to minimize the maximum weighted distance among all demand points and their assigned facilities. The p-center problems that we consider are discrete location problems, since the number of demand points that are served by the facilities is finite.

The main application areas of the p-center problem are emergency service locations such as fire and police stations, hospitals and ambulance services. In all of these application areas the main concern is to save human life. Hence, it is desirable to provide the service as quickly as possible by minimizing the farthest weighted distance of a demand point to a closest facility. Consider an ambulance service that has received an emergency call; the time spent during transportation is far more important than the cost of the transportation. Similarly, in the case of a fire, optimally placing a fire station for the quick delivery of an emergency service is more important than the cost of the delivery. Other application areas different from the emergency services could be considered as locating schools, industrial plants, warehouses, distribution centers and various service facilities in the telecommunication sector. Minimizing

the maximum coverage distance in the case of locating industrial plants and warehouses is crucial in terms of the delivery cost. As mentioned before, although the p-center problem enables to minimize the delivery cost of a business, the main concern in emergency service locations is to minimize the time of delivering a service.

For the most general form of the p-center problem, centers can be located anywhere on the network. The location of the centers may be at vertices or at any point on edges. The general version of the problem is referred to as the *absolute p-center problem*. The vertex-restricted case where the centers can only be placed at the vertices is referred to as the *vertex-restricted p-center problem*. If the problem is for only locating one center, then it is referred to as the *absolute 1-center* and the *vertex-restricted 1-center* problem, respectively. In practice, the absolute p-center problem is found more commonly since there is no restriction on locating the centers on the network, but this problem is considerably more complicated to solve relative to the vertex-restricted version. Most of the computational analysis and new algorithm developments are performed through the vertex restricted case of the p-center problem. On the contrary, the absolute p-center problem has received much less attention in the literature.

Hakimi [11] introduced the absolute 1-center problem by analyzing piecewise linear functions on each edge of the network. The minima of the piecewise linear functions on each edge are said to be local minimum points. The point with the smallest objective value among the local minimum points is the absolute 1-center of the network.

The absolute p-center problem for $p > 1$ is introduced again by Hakimi [12]. The problem is presented as follows:

Let $N = (V, E)$ be an undirected finite network with vertex set $V = \{ V_1, V_2, \dots V_n \}$ and edge set E consisting of undirected edges (V_i, V_j) for some index set I consisting of indices i, j for which $(V_i, V_j) \in E$. Let $d(V_i, V_j)$ be the length of a shortest path connecting V_i and V_j and $D(X, V_i) = \min_{x \in X} d(x, V_i)$ where X is a set of p points on the network.

Define the function f for $X \subset N$ by $f(X) = \max_{1 \le i \le n} w_i D(V_i, X)$ where the w_i are positive weights of V_i. The purpose of the p-center problem is to find an absolute p-center $X^* = \{X_1^*, X_2^* \ldots, X_p^*\}$ and the p-radius r_p for which $r_p \equiv f(X^*) = \min [\, f(X): |X| = p, X \subset N \,]$.

In the absolute p-center problem, X is a p-element subset of an undirected finite network N, and in the vertex-restricted p-center problem X is a p-element subset of V. In the case where all the weights w_i are equal to one, the problem is referred as *unweighted*; otherwise it is referred as the *weighted* case of the p-center problem.

In the vertex-restricted case, capacity bounds may also be associated with vertices in which case the problem is referred to as the capacitated p-center problem. The capacitated version of the absolute p-center problem has not been considered in the literature. In the capacitated vertex-restricted p-center problem each center must be located on a vertex such that the total weight of all assigned demand points to it cannot exceed the capacity of the vertex.

In this thesis, we study an algorithm and perform computational analysis on the p-center problem. The algorithm is proposed by Tansel [24] and focuses on solving the p-center problem via a finite series of minimum set covering problems which differ in their construction from the traditionally used set cover problems in the literature. The algorithm is applicable for both absolute and vertex-restricted p-center problems with weighted and unweighted cases. Furthermore, in order to compare the classical approach in the literature and the proposed method, we implement a software code for both of the algorithms. Note that, by the classical method, we mean the set-covering based method existing in the literature proposed initially by Minieka [18] for the unweighted case and extended by Kariv and Hakimi [17] to the weighted case (see also the implementation by Garfinkel, Neebe and Rao [10]). The first and only computational study prior to ours on the absolute p-center problem is given by Bozkaya and Tansel [2] based on test problems with up to 40 vertices. There are

no other computational studies on the absolute p-center problem subsequent to Bozkaya and Tansel [2] except our work in this thesis based on test problems with up to 900 vertices.

The proposed algorithm consists of two stages. In the first stage, we evaluate antipodals, intersection points and corresponding radius values on an undirected finite network. An antipodal is a unique point on an edge of a network at which the maximum value of a piecewise linear concave function is obtained. A point x on some edge of N is an intersection point if there exists distinct i and j such that $d(V_i, x) = d(x, V_j)$ and any movement from x by a small distance, say Є(Є>0), in any direction makes one of the distances $d(V_i, x)$ or $d(x, V_j)$ strictly larger. The radius value is a requirement that comes from the nature of the set covering and dominating set problems, which is commonly used in the solution procedures of the p-center problems. These concepts are explained in more detail in the forthcoming chapters. The intersection points and radius values are calculated by using antipodals. Each intersection point has its own radius value. These intersection point-radius pairs are considered as the candidate solutions for locating the centers on the network. In the second stage of the algorithm, a finite series of minimum set covering problems are solved by using the candidate solutions calculated in the first stage. These finite series of minimum set cover problems gives the minimum radius value and the locations of the centers for the p-center problem.

The forthcoming chapter gives a literature review of the p-center problem for both absolute and vertex-restricted cases. Chapter 3 introduces some notations and definitions that are used in the algorithm and gives a precise explanation of the algorithm used in this study. Additionally, an example is given in Chapter 3 with a sample network in order to illustrate the steps of the algorithm. Chapter 4 is devoted to the implementation of the algorithm. Computational analysis and numerical results are presented in Chapter 5. Finally, Chapter 6 concludes the thesis by giving a discussion on the results obtained from the proposed method and listing some future research topics.

Chapter 2

LITERATURE REVIEW

The p-center problem involves determination of locations of p facilities while minimizing the maximum weighted distance between demand points and facilities. Absolute 1-center problem is originally introduced by Hakimi [11]. Successively, a number of solution procedures are introduced. The distinctive feature of these solution procedures is that they focus on solving the p-center problem via a finite series of minimum set covering problems.

Minieka [18] proposed an algorithm for the unweighted absolute p-center problem that solves a finite number of set covering problems. Christofides and Viola [3] suggested an iterative algorithm for solving the weighted and unweighted absolute p-center problems and Garfinkel, Neebe, and Rao [10] gave an exact algorithm for the unweighted absolute p-center problem. All three algorithms are based on the set covering problem. More recently, Ilhan and Pınar [16] proposed a two stage LP - IP formulation where LP bounds are evaluated in the first phase and a number of set covering problems are solved in the second phase. The algorithm developed by Ilhan and Pınar [16] for the vertex-restricted p-center problem is modified and extended by Özsoy and Pınar [22] in order to solve the capacitated version of the p-center problem. Lastly, Elloumi, Labbe, and Pochet [9] introduced a new formulation for the vertex restricted p-center problem and obtained decent computational results.

Hakimi, Schmeichel, and Pierce [13] presented some improvements and generalizations for the 1-center problem proposed by Hakimi [11]. Minieka [19] extended the previous results on evaluating the absolute p-center problem. Kariv and Hakimi [17] proposed two different algorithms for the absolute 1-center and the absolute p-center problems with weighted and unweighted cases. Bozkaya and Tansel [2] focused on the spanning trees of the connected networks whose optimal p-center solution is the same as that of the connected network. Their motivation follows from the fact that the p-center problem is polynomially solvable on tree networks while it is NP-hard on general networks. Dvir and Handler [8] presented an algorithm for the unweighted absolute 1-center problem.

Tansel, Francis, and Lowe [23] examine the studies on the p-center problems on networks and provide 117 references on location problems on networks.

In the following sections of this chapter, the p-center literature is analyzed in detail under the absolute 1-center, the absolute p-center and the vertex restricted p-center problems.

2.1 The Absolute 1-Center Problem

Hakimi [11] introduced the absolute 1-center problem first by analyzing the piecewise linear functions on each edge of a network. The intersections of the piecewise linear functions where the slopes of the pieces are oppositely signed qualify as local minimum points. The smallest among the local minimum points is the absolute 1-center of the network.

Dearing and Francis [6] developed a formulation for the weighted absolute 1-center problem. They show that the radius of the problem is bounded below by:

$$max\left\{\left(d(V_i, V_j)\right)/\left((1/w_i) + (1/w_j)\right) : \forall_i, \forall_j \in N\right\}$$

Furthermore, they prove that the lower bound can always be attained for tree networks.

Minieka [20] presented an algorithm for the absolute 1-center problem on general networks. The algorithm requires only a knowledge of the shortest path distances between all pairs of vertices. For an edge on the network, initially the vertex set of the network is partitioned into two sets. The partitioned sets are identified according to the shortest path distances to the end points of the corresponding edge. First, center of all vertices is assumed to be one of the end points of the edge and the radius value is calculated. Next the farthest vertex to the selected end point is assigned to the other end point of the edge and accordingly the radius value is updated. The algorithm seeks until all vertices are assigned to the other end point which is initially not assumed to be the center. The point that results with the minimum radius value is the best local minimizer of the edge under consideration. The absolute center is selected among all local minimizers.

Dvir and Handler [8] presented an algorithm for the unweighted absolute 1-center problem. The worst-case complexities of the algorithm is $O(n^2 \log n + |E|n)$ where n and $|E|$ indicates number of vertices and the number of edges of the network, respectively. The algorithm is applicable for undirected networks and is based on the concept of minimum-diameter trees. The initial step of the algorithm is finding farthest nodes from each node to a vertex center of the network. Next step is to scan all edges of the network sequentially in an arbitrary order. During the scanning procedure, algorithm checks whether an edge contains an absolute-center or not by comparing with the lower bound values of the candidate absolute center of the corresponding edge.

Handler [14] propose an algorithm for the absolute 1-center problem on tree networks. The algorithm locates the absolute center at the midpoint of any longest path in the tree network. In order to find the longest path, the algorithm first selects an arbitrary vertex then finds the farthest vertex to the arbitrary selected vertex. The midpoint of the longest path is the unique absolute center of the tree network. The algorithm requires a computational effort of $O(n)$.

2.2 The Absolute p-Center Problem

The absolute p-center problem for $p > 1$ is initially introduced again by Hakimi [12].

An early algorithm proposed by Minieka [18] focuses on solving the p-center problem with a finite series of minimum set covering problem. Each set covering problem checks whether the clients can be covered within a threshold distance considered as radius, using p or fewer facilities.

Hakimi, Schmeichel, and Pierce [13] present improvements and generalizations for the 1-center problem proposed by Hakimi [12]. For a network with n vertices and $|E|$ edges, the complexity of finding the absolute 1-center is $O(|E|n^2 \log n)$ for the weighted case and $O(|E|n \log n)$ for the unweighted case. Furthermore they propose an algorithm for finding the unweighted absolute p-center of a tree network with a complexity of $O(n^{p-1})$.

Christofides and Viola [3] develop an iterative algorithm for computing the weighted and unweighted absolute p-centers. The algorithm initially finds regions in the network such that each region is covered by the same set of vertices. In other words, the regions are identified by whether they are reached by a vertex within a fixed radius value, say r, or not. After identifying the regions, a new network is formed with vertices consisting of the regions

and the initial vertex set of the network. The edges between regions and initial vertex set is formed if the points in the region can reach the vertex in the initial vertex set. Finally, a finite series of minimum set covering problem is solved until reaching p regions. The fixed radius value r is updated according to whether the solution of the minimum set cover problem is larger than p or not.

Garfinkel, Neebe, and Rao [10] construct an exact algorithm for the unweighted absolute p-center problem. The algorithm is based on the algorithm proposed by Minieka [18]. The major advantage of this algorithm in comparison to the one proposed by Minieka [18] is that it reduces the size of the problem which in turn reduces the computational time.

Minieka [19] extend the previous results for evaluating the absolute p-center problem and conclude that various absolute p-center problems could be solved by the same techniques used in the vertex restricted case of the p-center problem. The only difference is to use edge distance functions instead of vertex distance functions.

Kariv and Hakimi [17] propose two different algorithms for the absolute 1-center problem and the absolute p-center problem with weighted and unweighted vertices. The computational complexities of the algorithms are as follows: weighted absolute 1-center is $O(|E|n\log n)$; unweighted absolute 1-center is $O(|E|n+\mathrm{n}^2\log n)$; weighted absolute p-center is $O(|E|^p(n^{2p-1})\ /\ (p-1)!\log n)$; unweighted absolute p-center is $O(|E|^p(n^{2p-1})\ /\ (p-1)!)$. In addition to these, they propose algorithms for both 1-center and p-center problems on tree networks for both weighted and unweighted cases. Furthermore, they show that the p-center problem on general networks is NP-hard [17].

Bozkaya and Tansel [2] focus on spanning trees of any connected network whose optimal p-center solution is same as that of the connected network. In other words; the consideration is about the search strategy types in order to find spanning tree structures whose optimal p-center solutions are the same as those of the connected networks. In order to search

and explore these spanning tree structures, Bozkaya and Tansel [2] first prove the existence of an “optimal” spanning tree whose optimal p-center solution is equal to the optimal p-center solution of the connected network. The proof requires a knowledge of the p-center of a network to construct such a spanning tree. Bozkaya and Tansel [2] introduce two types of trees that may contain an optimal tree. Both of the trees are considered as rooted shortest path trees. Trees are constructed by picking certain points of the network as roots and forming the union of shortest paths that connect the roots to the vertices. The first type of tree that is considered has roots at the segments that are defined by adjacent elements of the vertex set which has antipodals on vertices. The second type of trees are rooted at intersection points of an undirected network. In addition to these they made computational tests up to n = 40 nodes in order to see if these two sets contain optimal spanning tree.

2.3 The Vertex-Restricted p-Center Problem

The algorithm proposed by Ilhan and Pınar [16] for the vertex-restricted case could be considered as a two stage LP - IP formulation, where in the first stage, the LP relaxation is solved by relaxing the total number of facilities to be located. This provides a suitable lower bound on the optimal value. In the second stage, the IP formulation is solved by starting from the lower bound obtained in the first stage. In the IP formulation of the second stage, the lower bound found in the first stage is increased until a feasible IP solution is obtained. The algorithm initializes by selecting initial upper and lower bounds on the objective function value and setting the coverage distance to the average of the lower and upper bounds. After solving an appropriate set covering problem based on this coverage distance, the upper bound is replaced with the coverage distance if it is possible to cover all clients with at most p facilities. If it is not possible to cover all clients with at most p facilities, then the lower bound is replaced with the coverage distance. This procedure continues until upper and lower bounds are equal to each other. The appropriate set covering problems that are solved in the algorithm

have the objective of minimizing the number of facilities to be located while covering all clients. Although the algorithm proposed by Ilhan and Pınar [16] has two stages with LP and IP formulations, the algorithm is computationally efficient. At each iteration, the proposed algorithm sets a threshold distance as a radius to see whether it is possible to cover all clients with p or less facilities within this radius. Up to now the algorithm works the same as the algorithm that Minieka [18] proposed except that this modified algorithm updates lower and upper bounds on the optimal radius which results in less computational effort. The importance of the paper proposed by Ilhan and Pınar [16] is that it is the first paper that provides an optimal solution to a large scale vertex-restricted p-center problem. The sizes of the problems during the computational analysis range from 100 nodes to 900 nodes.

The algorithm developed by Ilhan and Pınar [16] for the vertex-restricted p-center problem is modified and extended by Özsoy and Pınar [22] for the case of the capacitated p-center problem. The algorithm proposed by Özsoy and Pınar [22] has two phases similar to the algorithm of Ilhan and Pınar [16]. In the first phase, LP relaxations of the sub-problems are solved to carry out a binary search over the distance values in order to provide a suitable starting point for the search in Phase 2.

Elloumi, Labbe, and Pochet [9] introduce a new formulation for the vertex restricted p-center problem that is based on solving a set-covering problem. The model aims to find the maximum coverage distance such that all clients are covered within that distance with at most p centers. The new formulation has been compared with the previously developed vertex-restricted p-center problem formulations. Elloumi, Labbe, and Pochet [9] present a polynomial time algorithm for computing lower and upper bounds of the optimal solution. According to the results obtained in [9], the proposed formulation gives better lower bound values than the previously suggested vertex-restricted p-center algorithms. Furthermore they show that the lower bound obtained from the algorithm cannot be less than one third of the optimal radius when distances satisfy triangle inequalities. The computational experiments show that the lower bounds are slightly better than the ones obtained in the first phase of Ilhan

and Pinar [16] algorithm. The sizes of the problems during the computational analysis range from 100 nodes to 1817 nodes.

Daskin [4] propose an algorithm which is based on the idea developed by Minieka [18]. The purpose of the formulation is based on decreasing the difference between the upper and lower bounds of the optimal solution for the vertex p-center problem. The algorithm arbitrarily selects some lower and upper bounds and sets the average of them as the radius value for the set covering problem to be solved. If the optimal value of the related set covering problem is less than or equal to p, the upper bound value and accordingly the radius value is updated and the optimal solution value of the set covering problem is obtained. If the optimal value of the set covering problem is greater than p, the lower bound and accordingly the radius value is updated. The algorithm terminates when the gap between the lower and upper bounds is equal to zero. Daskin [5] improves the algorithm proposed by Daskin [4] by formulating a maximal set covering subproblem and solving it using a Lagrangian relaxation method.

Relaxation could be considered as a method of solving large scale location problems by using small sized sub-problems. The relaxation algorithm is an iterative approach which updates the bounds on the optimal solution at each step until the optimal solution is obtained. Handler and Mirchandani [15] develop an iterative relaxation algorithm to solve the p-center problem for a subset of demand points. The algorithm initially finds a feasible solution to the relaxed problem. Next, the algorithm checks the feasibility of the solution to the original problem. If the original problem is not covered, a point farthest from its closest center is added to the relaxed problem and the procedure is repeated. If the original problem is covered, the procedure looks for a better solution than the current one. The algorithm terminates when there is no better feasible solution to the relaxation problem.

Al-khedhairi and Salhi [1] propose modifications for the algorithms by Ilhan and Pınar [16] and Daskin [4]. They suggest two improvements to Ilhan and Pınar's [16] algorithm: In the first one, instead of setting the radius value equal to the lower bound when the LP relaxation is feasible, keeping it unchanged decreases the number of iterations that needs to be performed. The second improvement is based on the fact that the coverage radius will take a value from the distance matrix. Being aware of this fact, one can check if the radius value to solve the IP feasibility problem is in the distance matrix and select the next minimum distance, which is greater than the current one, as the radius value if it is not in the distance matrix. For Daskin's algorithm [4], they initially suggest some ideas to tighten the initial lower and upper bounds. In addition to that, they define some enhancement based on the golden section method. The computational experiments show that the enhancements decrease the number of iterations needed and decreases the computational time of the algorithms, reasonably.

Mirchandani and Francis [21] (Chapter 7 by Handler) propose a relaxation algorithm that finds the optimal location of centers by using column generation and set covering approaches. The algorithm initiates by selecting an arbitrary vertex. Next the algorithm identifies whether the arbitrarily selected vertex would cover all demand points within a specified radius value of r. If this is the case, then it is said to be a candidate center. Note that if the problem is for locating p number of centers, the procedure should be repeated p times. After identification of candidate centers, the candidate centers whose ranges exceed or equal to specified value r are eliminated. The elimination process discards the columns of shortest path distance matrix between candidate center points and vertices. The algorithm finally solves a set covering problem in order to optimally locate the p-centers.

Chapter 3

ALGORITHM & MATHEMATICAL FORMULATION

In practice, mostly the absolute p-center problem occurs since in real life applications centers can be located anywhere on the network but this problem is considerably more complicated to solve in comparison to the vertex-restricted version. Most of the computational analysis and new algorithm developments are performed on vertex restricted case of the p-center problem. On the contrary, the absolute p-center problem has received much less attention than the vertex restricted version.

As mentioned before, a common feature of the p-center problem solution procedures in the literature is that they focus on solving the p-center problem via a finite series of minimum set covering problems. In this chapter, we present an algorithm which focuses on solving the p-center problem with a similar approach based on the shortest path trees. We study the p-center problem for both absolute and vertex restricted cases but our main concern is on the absolute p-center problem due to the fact that it has received considerably less attention in the literature in terms of algorithmic and computational analysis.

The proposed algorithm consists of two stages. The first stage could be considered as a preprocessing stage where the antipodals and intersection point-radius pairs are calculated. In the second stage; a finite series of minimum set covering problems is solved by using the parameters evaluated in the first stage. It is convenient at this point to introduce some notations and definitions that are used in the algorithm.

3.1 Notations of the Algorithm

- $N = (V, E)$ represents an undirected finite network with vertex set $V = \{ V_1, V_2, \ldots V_n \}$ and edge set E consisting of undirected edges (V_i, V_j) for some index set I consisting of vertex indices i, j for which $(V_i, V_j) \in E$.
- $d(V_i, V_j)$ is the length of a shortest path from vertex V_i to vertex V_j.
- w_i represents the weight for vertex i.
- X_j represents the j^{th} intersection point.
- Y_i^{pq} represents an antipodal on edge (V_p, V_q), $p < q$, induced by vertex V_i.
- θ_i^{pq} is the distance between an antipodal and the end vertex V_p of the edge (V_p, V_q).
- λ_{ij}^{pq} is the distance between the end vertex V_p and the intersection point induced by vertices V_i and V_j on the edge (V_p, V_q).
- r_{ij}^{pq} is the radius value for the intersection point corresponding to λ_{ij}^{pq}.
- $X(\lambda_{ij}^{pq}, r_{ij}^{pq})$ represents an intersection point-radius pair which is induced by pair of vertices V_i and V_j on the edge (V_p, V_q).
- Y_{kl}^{pq} represents the intersection point defined by the pair of vertices V_k and V_l on the edge (V_p, V_q).

3.2 Definitions

3.1.1 Distance Function of a Point on a Given Edge

Let $[V_p, V_q]$ be an edge on network N and let x be any interior point of the edge $[V_p, V_q]$. We assume that N is connected and the edge lengths are all positive. A shortest path from x to a fixed vertex V_i must necessarily visit one of the end vertices V_p or V_q. If it visits V_p then it must follow a shortest path from V_p to V_i from that point on. Likewise, if it visits V_q then it must follow a shortest path from V_q to V_i from that point on. Hence the shortest path from any point x on edge $[V_p, V_q]$ to a fixed vertex V_i is given by the following function:
$d(x, V_i) = min\{\propto + d_{pi}, l_{pq} - \propto + d_{qi}\}$ where $d(x, V_i)$ is the shortest path distance function of x, l_{pq} is the length of edge $[V_p, V_q]$, $\propto$ is the length of the subedge $[V_p, x]$, d_{pi} and d_{qi} are the shortest path distances between V_p to V_i and V_q to V_i, respectively. The situation is illustrated in Figure 1.

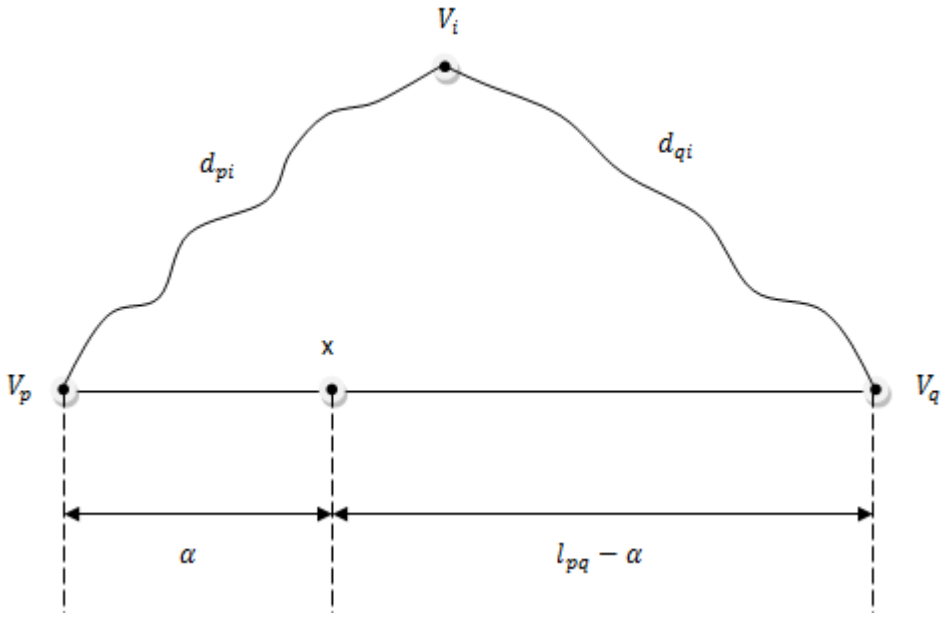


Figure 1 The illustration of shortest path distances

Observe that $\propto +d_{pi}$ is a linear function with slope +1 and that $l_{pq}-\propto +d_{qi}$ is a linear function with slope -1. The functions are illustrated in Figure 2.

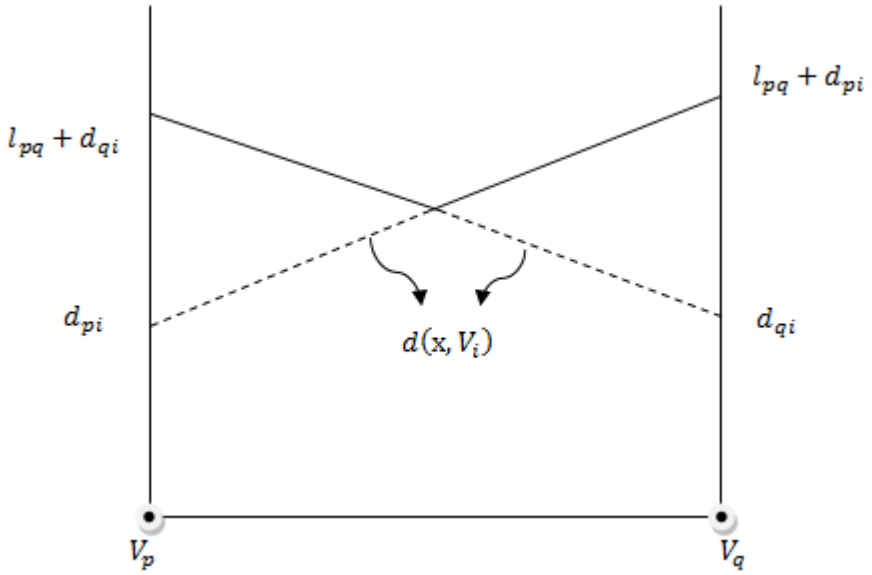


Figure 2 The illustration of shortest path distance functions

The shortest path distance function of x, $d(x, V_i)$, is a pointwise minimum of two linear functions. Accordingly, $d(x, V_i)$ is a piecewise linear concave function with at most two pieces.

3.1.2 Definition of Intersection Point

We say a vertex pair V_i, V_j induces an intersection point x on edge $[V_p, V_q]$ if $w_i\, d(V_i, x) = w_j\, d(V_j, x)$ with either $d(x, V_i) = d(V_i, V_p) + \propto$ and $d(x, V_j) = d(V_j, V_q) + l_{pq} - \propto$ or $d(x, V_j) = d(V_j, V_p) + \propto$ and $d(x, V_i) = d(V_i, V_q) + l_{pq} - \propto$. Here, $\propto$ is the length of the subedge with end points V_p and x. Note that if x is an intersection point induced by the vertex pair V_i and V_j, then x is a local minimum of the function $f_{ij}(x) \equiv max\{w_i\, d(V_i, x), w_j\, d(V_j, x)\}$. Figure 3 illustrates the intersection point x induced by V_i and V_j for the case $w_i = w_j = 1$.

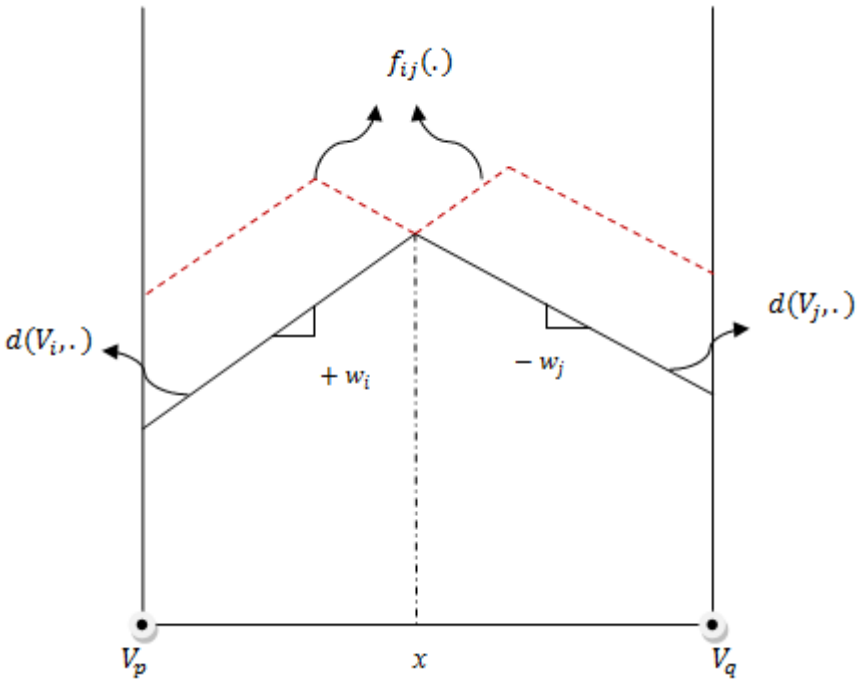


Figure 3 The illustration of an intersection point

The two-piece linear function with slopes $\pm w_j$ is the weighted distance function $w_j\, d(x, V_j)$ associated with vertex V_j while the two-piece linear function with slopes $\pm w_i$ is the weighted distance function $w_i\, d(x, V_i)$ associated with vertex V_i. The pointwise maximum of these functions is the portion with broken lines as illustrated in Figure 3.

3.1.3 Definition of Radius

The radius value is a parameter that is most commonly used in the solution procedures of set covering and p-center problems. Whenever a point x on an edge $[V_p, V_q]$ is an intersection point, there is a pair of distinct vertices V_i, V_j that induces it and weighted distances $w_i\, d(x, V_i)$ and $w_j\, d(x, V_j)$ are equal to each other. We define $r_{ij}^{pq} = w_i\, d(x, V_i) = w_j\, d(x, V_j)$ as the radius value of the intersection point induced by vertices V_i and V_j. In addition, we refer to the length of the edge segment between the intersection point and the end vertex V_p of edge $[V_p, V_q]$ as λ_{ij}^{pq}. Since every intersection point has its own radius value, we consider candidate center locations as intersection point-radius pairs and denote them as $X(\lambda_{ij}^{pq}, r_{ij}^{pq})$. An intersection point-radius pair is illustrated in Figure 4.

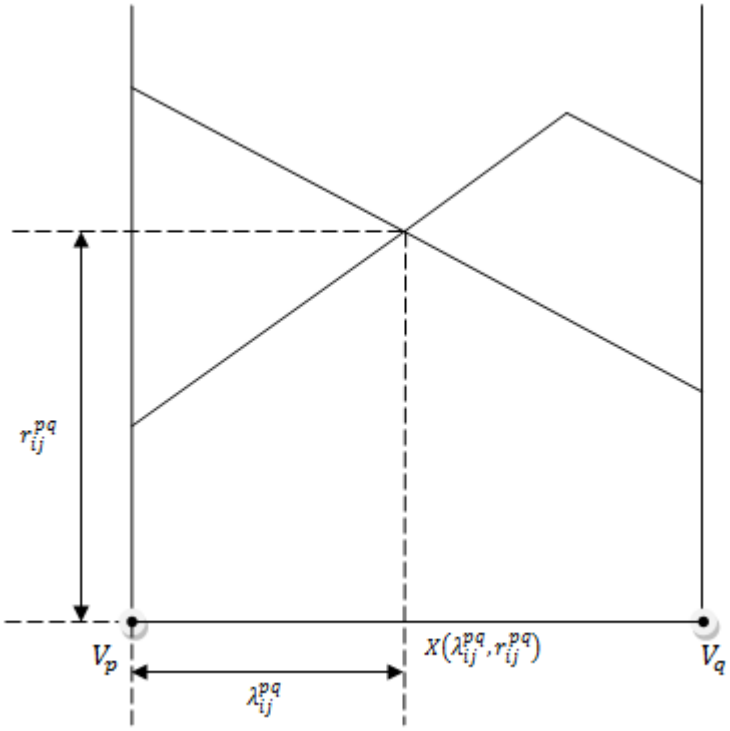


Figure 4 Illustration of an intersection point-radius pair

3.1.4 Definition of an Antipodal

Consider a vertex V_i and a variable point x on edge $[V_p, V_q]$. If there is a point y in the interior of $[V_p, V_q]$ such that $d(V_i, V_p) + \propto = d(V_i, V_q) + l_{pq} - \propto$ where $\propto$ is the length of the subedge $[V_p, y]$, then y is called an antipodal of V_i on edge $[V_p, V_q]$. Antipodals are denoted by Y_i^{pq} where i is the index of vertex V_i which induces an antipodal and p, q represents the edge where the antipodal is observed.

The distance between an antipodal to vertex V_p is denoted as θ_i^{pq}. An illustration of an antipodal is on Figure 5.

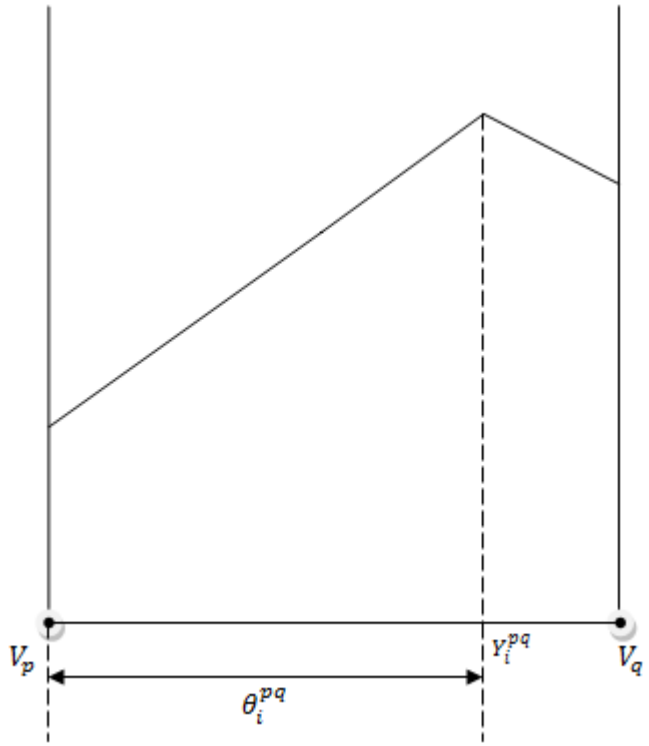
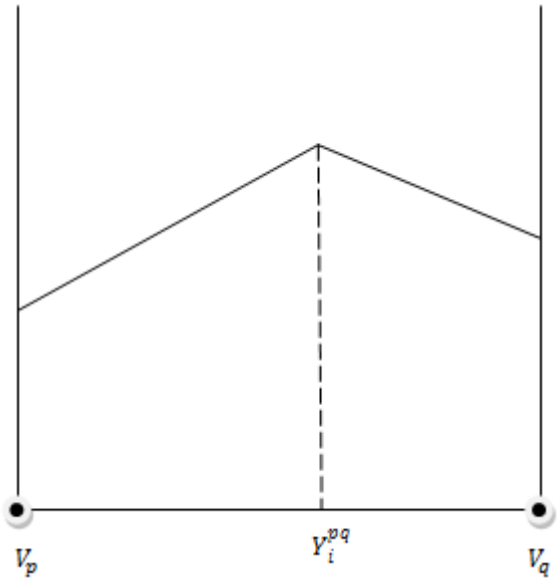


Figure 5 The illustration of an antipodal

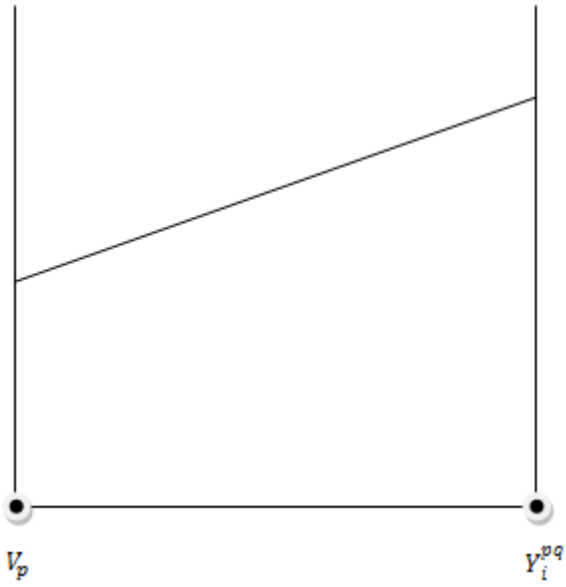
The motivation of finding antipodals in a network follows from the fact that the use of antipodals enables us to compute intersection point-radius pairs more efficiently and conveniently. In most cases there are piecewise linear functions on the edges of a network. Our aim is to find antipodals and convert these piecewise linear functions into linear functions. After the conversion, intersection point-radius pairs are easily calculated.

Consider the edge $[V_p, V_q]$ of the network N. Let Y_i^{pq} be an antipodal of edge $[V_p, V_q]$ induced by vertex V_i. We treat the edge segments in-between adjacent antipodals and between antipodals and end vertices as newly formed edges. Between newly formed edges there are only linear functions which enable ease of calculation for intersection point-radius pairs.

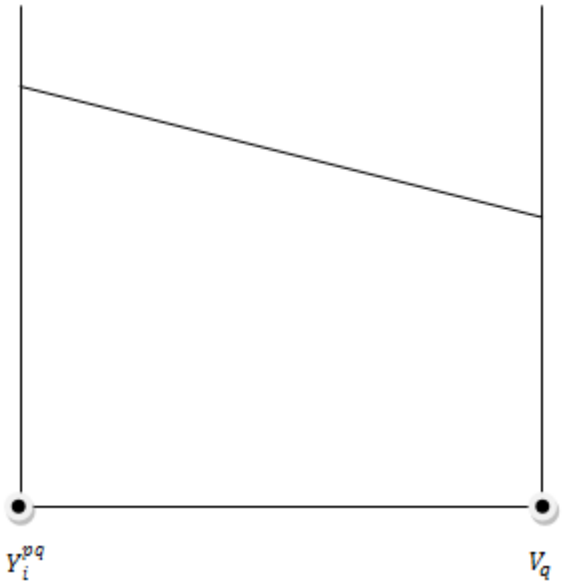
Figure 6a illustrates an antipodal on edge $[V_p, V_q]$ whereas Figure 6b and Figure 6c illustrates the linear distance functions of newly formed edges $[V_p, Y_i^{pq}]$ and $[Y_i^{pq}, V_q]$ respectively.



(a) Illustration of antipodal on edge $[V_p, V_q]$



(b) Distance function on edge $[V_p, Y_i^{pq}]$



(c) Distance function on edge $[Y_i^{pq}, V_q]$

Figure 6 The illustration of an antipodal and corresponding distance functions

3.3 Algorithm for Computing p-Center Solution

The first stage of the proposed algorithm is a preprocessing stage where the antipodals and intersection point-radius pairs are calculated. Next, shortest path trees that are rooted at intersection points are formed. Rooted shortest path trees are initially introduced in Bozkaya and Tansel [2]. In Bozkaya and Tansel [2], a shortest path tree is constructed by taking each intersection point to be the root and constructing a shortest path tree via Dijkastra's method [7] that connects the root to all vertices in the network. The consideration of using intersection points is because of the fact that a p-center of a network induces a partitioning of V and the network itself, which is closely related with considering intersection points as potential facility locations [2]. In the proposed method, shortest path trees are still rooted at intersection points, but they do not span all vertices in N. Instead, a shortest path tree spans only those vertices whose weighted distances to the root are within the radius value of the intersection point-radius pair.

Let T_{ij}^{x} be the shortest path tree rooted at x that includes all vertices V_k for which $W_k\, d(x, V_k) \leq r_{ij}^{pq}$. Note that if an intersection point-radius pair induces more than one such tree corresponding to different pairs of vertices with different radius values, these trees are treated as distinct trees.

Let $T_1, T_2, \dots, T_l$ be an enumeration of all such trees with associated radius values $r_1, r_2, \dots, r_l$. We say T_i and T_j intersect if there is at least one vertex included in both, otherwise they do not intersect. Associated with each tree T_j, define a zero/one variable X_j that is going to be one if the intersection point-radius pair that defines T_j is selected as a center and zero if it is not selected as a center. We want to choose p of the trees so that the chosen ones cover the vertex set and largest radius value associated with selected trees is minimum.

More formally, the algorithm is stated step by step as follows:

Step 1a) Compute the antipodal of every vertex on every edge (if it exists) and extend the definition of the vertex set to include the antipodals and extend the definition of the edge set to include newly formed edges defined by adjacent pairs of vertices in the extended vertex set.

Step 1b) Compute the intersection point between each vertex pair on each edge of the extended network.

Step 1c) Compute the radius value r_{ij}^{pq} via $r_{ij}^{pq} = w_i\, d(x, V_i) = w_j\, d(x, V_j)$ where V_i and V_j are the vertices that define the intersection point x. Call $X(\lambda_{ij}^{pq}, r_{ij}^{pq})$ the intersection point-radius pairs of the network and define the radius set associated with $r_1, \dots, r_m$ where m is the number of distinct radius values.

Step 2) List the λ_{ij}^{pq} and r_{ij}^{pq} values computed in steps 1a, 1b and 1c. Browsing through the list, form the rooted subtrees associated with the intersection point-radius pairs $X(\lambda_{ij}^{pq}, r_{ij}^{pq})$. Note that the subtrees are formed for each $[\lambda_{ij}^{pq}, r_{ij}^{pq}]$ pair computed in steps 1b and 1c. The trees are constructed by picking intersection point-radius pairs of the network as roots and by forming the shortest path trees via Dijkstra's algorithm that connects the root to the vertices reached within the radii of the intersection points. That is, the node set $N(\lambda_{ij}^{pq}, r_{ij}^{pq})$ of a tree $T(\lambda_{ij}^{pq}, r_{ij}^{pq})$ is defined as $N(\lambda_{ij}^{pq}, r_{ij}^{pq}) = \{V_k \in N: d(V_k, x) \leq r_{ij}^{pq}/w_k\}$. Let these trees be enumerated as $T_1, T_2, \dots, T_l$ with radii $r_1, r_2, \dots, r_l$ where l is the number of distinct intersection point-radius pairs of the network.

Step 3) Associated with each tree T_j define a zero/one variable X_j that is going to be one if the intersection point-radius pair that defines T_j is selected as a center and zero if not.

Step 4) Solve a finite series of minimum set covering problems so that p of the trees cover the vertex set and the largest radius value associated with the selected set of trees is minimum. Note that the union of the trees must contain or span all of the vertices in the network N.

The minimum set covering model that is being used in the algorithm is as follows:

Let

$$X_j = \begin{cases} 1 \quad if\ the\ intersection\ point\ that\ defines\ the\ tree\ T_j\ is\ selected\ as\ a\ center \\ 0\ otherwise \end{cases}$$

$$a_{ij} = \begin{cases} 1 \quad if\ vertex\ V_i\ is\ included\ in\ the\ tree\ T_j \quad i.e\ \ w_i\ d(V_i, X_j) \leq r_j \\ 0\ otherwise \end{cases}$$

Define $J(r)$ to be the set of tree indices j such that the radius r_j of the tree T_j is no larger than a given radius value r.

$$q(r) = \min \sum_{j \in J(r)} X_j \qquad (1)$$

s.t

$$\sum_{j \in J(r)} a_{ij} X_j \geq 1 \qquad i \in \{1, \dots, n\} \qquad (2)$$

$$X_j \in \{0,1\} \qquad j \in J(r) \qquad (3)$$

(1) provides the number of selected centers to be minimum, (2) enables every vertex V_i to be included in some tree. (3) is the binary definition of the variables X_j.

At each set covering problem, the model finds a positive integer $q(r)$ as the objective function value indicating the minimum number of centers to be located in the network while covering all vertices within a weighted distance of at most r. Moreover, the model performs a binary search on the ordered set of radius values of the network. The model seeks for the smallest radius value r in the finite set $\{r_1, \dots, r_m\}$ whose objective value is less than or equal to p.

If $q(r) > p$ then we need to increase the value of the radius r since more than p points are needed to cover all vertices within a radius value of r. We increase the radius value with the binary search on the radius set of the network. If $q(r) \leq p$ then we need to decrease the value of the radius r since this radius value r permits to place at most p centers on the network and still cover all vertices within a radius value of r. We stop with the smallest radius value r for which $q(r) \leq p$.

3.4 Illustration of the Algorithm with an Example

In order to characterize the steps of the algorithm more clearly, we considered a sample network that is demonstrated by Hakimi [11]. The network consists of six vertices and eight undirected edges. The network is illustrated in Figure 7.

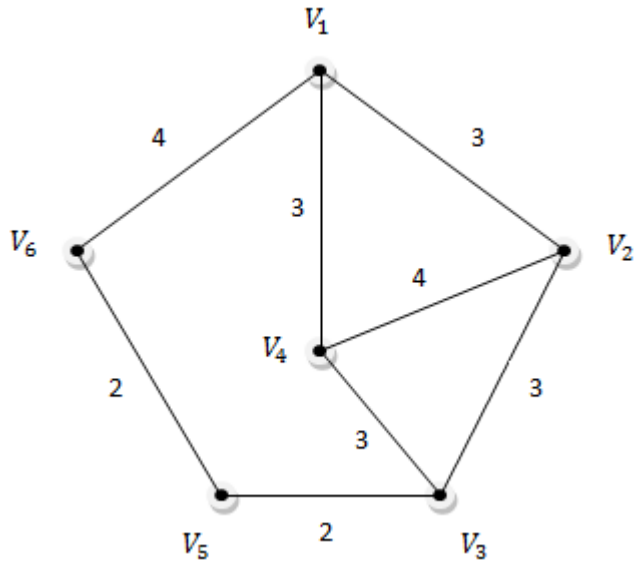


Figure 7 The illustration of the network

Vertex Set $= \{ V_1, V_2, V_3, V_4, V_5, V_6 \}$

Edge Set $= \{ [V_1, V_6], [V_5, V_6], [V_5, V_3], [V_3, V_4], [V_3, V_2], [V_2, V_4], [V_1, V_4], [V_1, V_2] \}$

In the network illustrated in Figure 7, edge lengths are shown next to the edges and all vertex weights are assumed to be one.

Initially, shortest path distances between each pair of vertices are evaluated via Dijkastra's shortest path algorithm. The shortest distance matrix of the network is as follows:

$$\begin{pmatrix} 0 & 3 & 6 & 3 & 6 & 4 \\ 3 & 0 & 3 & 4 & 5 & 7 \\ 6 & 3 & 0 & 3 & 2 & 4 \\ 3 & 4 & 3 & 0 & 5 & 7 \\ 6 & 5 & 2 & 5 & 0 & 2 \\ 4 & 7 & 4 & 7 & 2 & 0 \end{pmatrix}$$

Evaluation of Antipodals

Let $[V_p, V_q]$ be an edge and let V_i be any vertex of the network N. Antipodals are evaluated by checking the following two conditions:

1) $d_{pi} + l_{pq} > d_{qi}$
2) $d_{qi} + l_{pq} > d_{ip}$

If these conditions are both satisfied then this implies that there exist an antipodal of V_i on edge $[V_p, V_q]$ otherwise there is no antipodal of V_i on edge $[V_p, V_q]$. The above conditions imply that there is a piecewise linear function on the edge $[V_p, V_q]$ and there is a unique point Y_i^{pq} where the piecewise linear function attains its maximum value. Y_i^{pq} is the antipodal of V_i. The above conditions and explanations are illustrated in Figure 8.

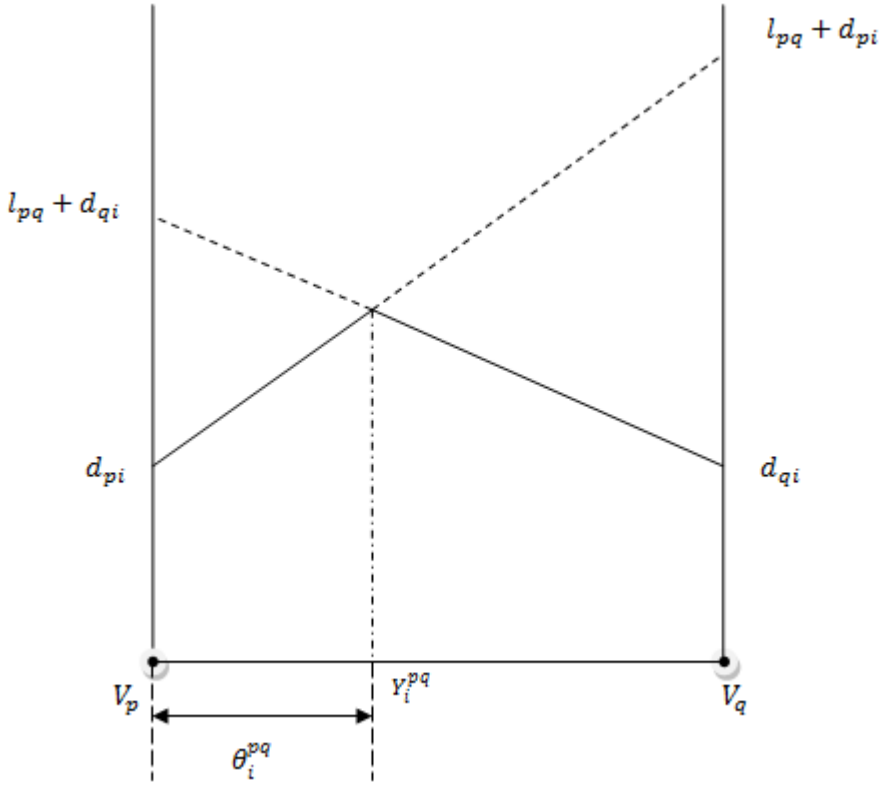


Figure 8 The illustration of the conditions for evaluating antipodal

After checking the conditions through all edges and vertices; we need to calculate the distance between the antipodal and end vertex of the corresponding edge. The distance between an antipodal and vertex V_p is denoted as θ_i^{pq} and calculated as follows:

$$\theta_i^{pq} = \frac{1}{2}(d_{qi} + l_{pq} - d_{ip})$$

The measure of θ_i^{pq} enables us to identify the location of the antipodal on the specified edge and it is illustrated in Figure 9.

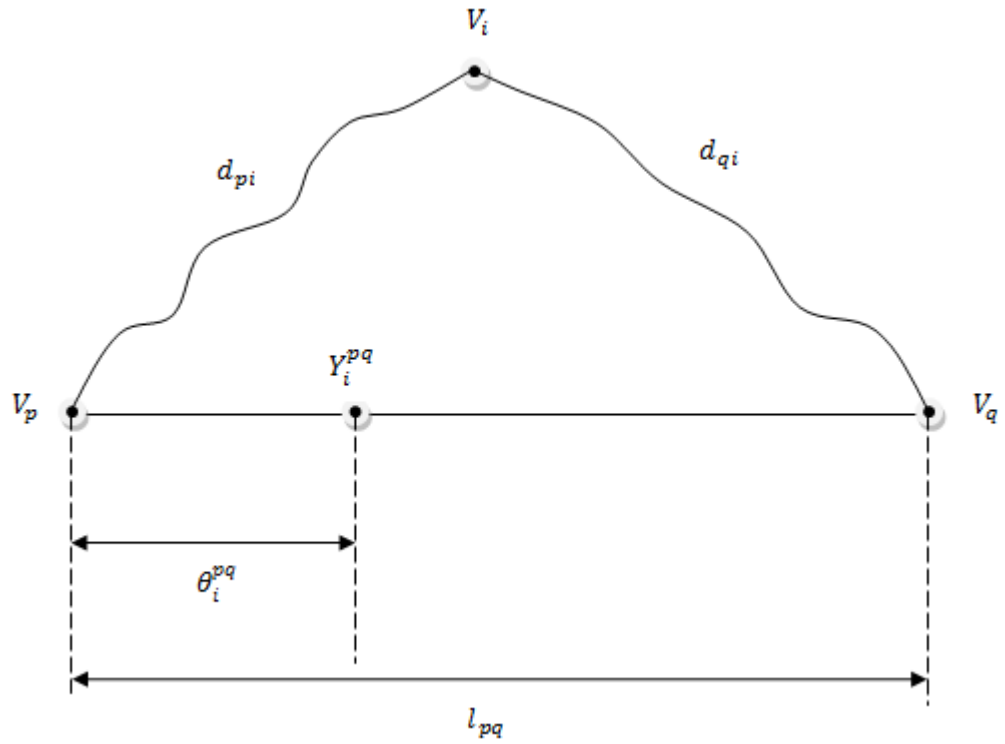


Figure 9 The illustration of an antipodal on an edge

The antipodals of the example are calculated by checking the previously mentioned two conditions and listed as follows:

Y_i^{pq}	V_i	$[V_p, V_q]$	θ_i^{pq}
Y_1^{24}	1	[2,4]	2
Y_1^{35}	1	[3,5]	1
Y_2^{14}	2	[1,4]	2
Y_2^{34}	2	[3,4]	2
Y_3^{16}	3	[1,6]	1
Y_3^{24}	3	[2,4]	2
Y_4^{12}	4	[1,2]	2
Y_4^{23}	4	[2,3]	1
Y_5^{12}	5	[1,2]	1
Y_5^{14}	5	[1,4]	1
Y_5^{24}	5	[2,4]	2
Y_6^{24}	6	[2,4]	2

The first column represents the identity of the antipodal where the upper index is the edge of the antipodal and the lower index i is the index of the vertex V_i that induces the antipodal. The second column lists vertices V_i whereas the third column lists the corresponding edges of the antipodals. The distance between an antipodal to vertex V_p is denoted as θ_i^{pq} and presented in the fourth column.

Recall that, we define the edge portions in-between adjacent antipodals and between antipodals and end vertices as newly formed edges. The newly formed edges of the example is illustrated in Figure 10.

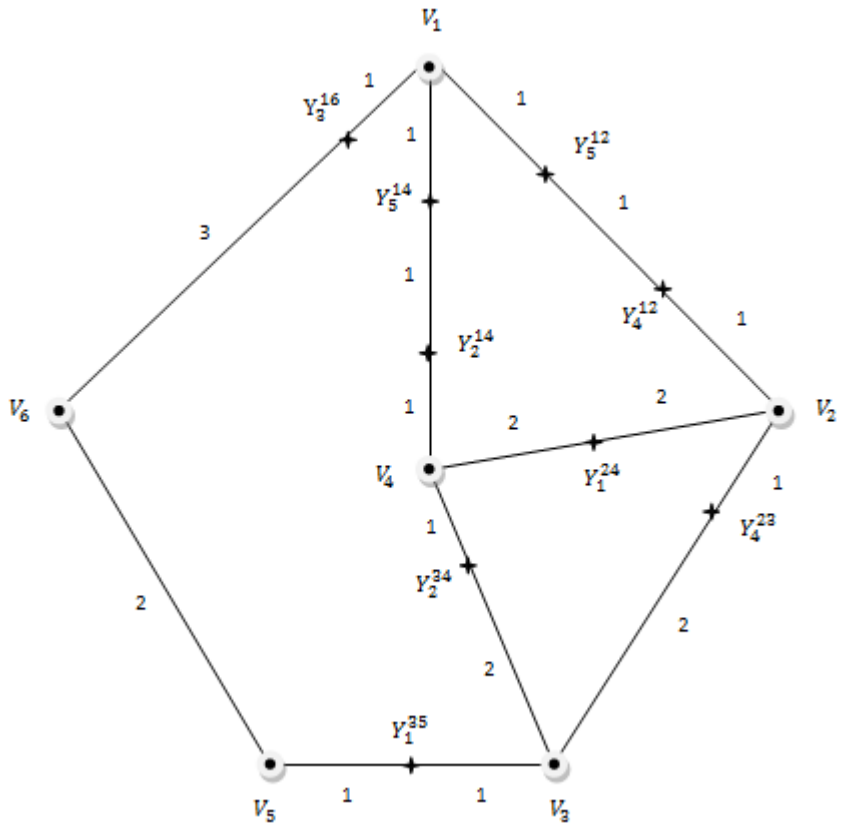


Figure 10 Illustration of newly formed edges on the network

Extended Vertex Set

$\{ V_1, V_2, V_3, V_4, V_5, V_6, Y_1^{24}, Y_1^{35}, Y_2^{14}, Y_2^{34}, Y_3^{16}, Y_4^{12}, Y_4^{23}, Y_5^{12}, Y_5^{14} \}$

Newly Formed Edge Set

$[V_1\ Y_3^{16}], [Y_3^{16}\ V_6], [V_5\ Y_1^{35}], [Y_1^{35} V_3], [V_3\ Y_2^{34}], [Y_2^{34} V_4], [V_3\ Y_4^{23}], [Y_4^{23}\ V_2], [V_2\ Y_1^{24}], [Y_1^{24}\ V_4], [V_1\ Y_5^{14}], [Y_5^{14}\ Y_2^{14}], [Y_2^{14}\ V_4], [V_1\ Y_5^{12}], [Y_5^{12}\ Y_4^{12}], [Y_4^{12}\ V_2], [V_5\ V_6]$

An intersection point *x* could also be defined as a point on any edge defined by two distinct vertices V_k and V_l such that the distance functions of the vertices intersect at *x*, one with a positive slope and the other with a negative slope. (Figure 11)

Within the newly formed edge set, instead of piecewise linear distance functions, there are only linear distance functions. The intersection points are easily calculated by using the newly formed edges. If two intersecting linear functions have oppositely signed slopes within a newly formed edge, then an intersection point occurs.

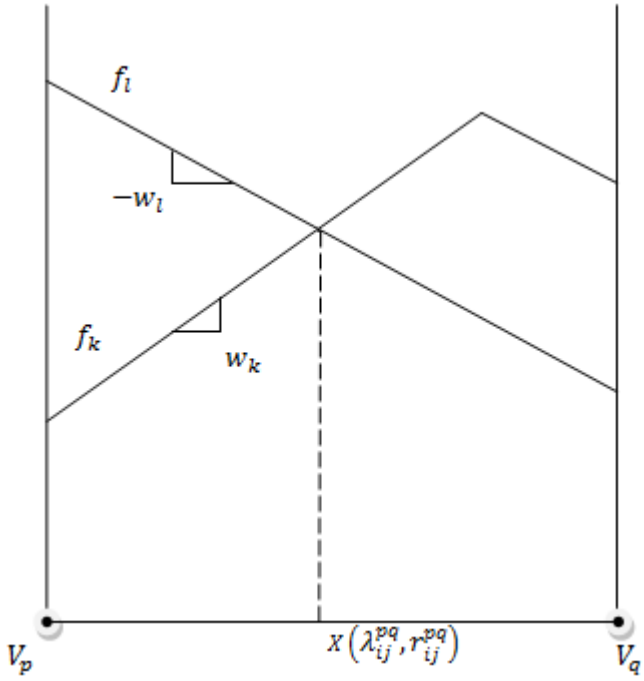


Figure 11 Illustration of an intersection point by piecewise linear distance functions

Evaluation of Intersection Points

After calculating antipodals and forming the new edge set, intersection points are evaluated by using the vertex set and newly formed edge set listed previously. If we were to calculate the intersection points without calculating antipodals, the search and evaluation of intersection points would be more complicated since we were going to deal with piecewise linear functions on each edge of the network.

Recall that the weighted distance functions are defined as $f_k(x) = w_k\, d(V_k, x)$ and $f_l(x) = w_l\, d(V_l, x)$. If functions $f_k(x)$ and $f_l(x)$ intersect with oppositely signed slopes, then this implies that there exist an intersection point at point *x*. Note that, for the unweighted case of the problem the weights w_k and w_l are equal to one hence the slopes of the functions $f_k(x)$ and $f_l(x)$ would be +1 and -1 respectively. In order to detect the functions that intersect on each edge, let I_1^{pq} be the set of indices that have increasing distance function $f_k(.)$ on a newly formed edge and let I_2^{pq} be the set of indices that have decreasing distance function $f_l(.)$ on the same newly formed edge segment. Clearly, $I_1^{pq} \cup I_2^{pq} = I \equiv \{1, \dots, n\}$ and $I_1^{pq} \cap I_2^{pq} = \emptyset$.

We calculate the intersection point-radius pairs by checking the following conditions on each newly formed edge segments:

1) $w_k\, d\left(V_k, Y_j^{pq}\right) \leq w_l\, d\left(V_l, Y_j^{pq}\right)$
2) $w_k\, d\left(V_k, Y_{j+1}^{pq}\right) \geq w_l\, d\left(V_l, Y_{j+1}^{pq}\right)$

If the conditions are satisfied, then there is an intersection point on this edge induced by vertices V_k and V_l and it is found by solving $w_k \left[d\left(V_k, Y_j^{pq}\right) + \lambda_{kl}\right] = w_l\left[d\left(V_l, Y_{j+1}^{pq}\right) + l - \lambda_{kl}\right]$ where l is the length of the edge defined by Y_j^{pq} and Y_{j+1}^{pq}. Figure 12 illustrates the intersection point when the above conditions are satisfied.

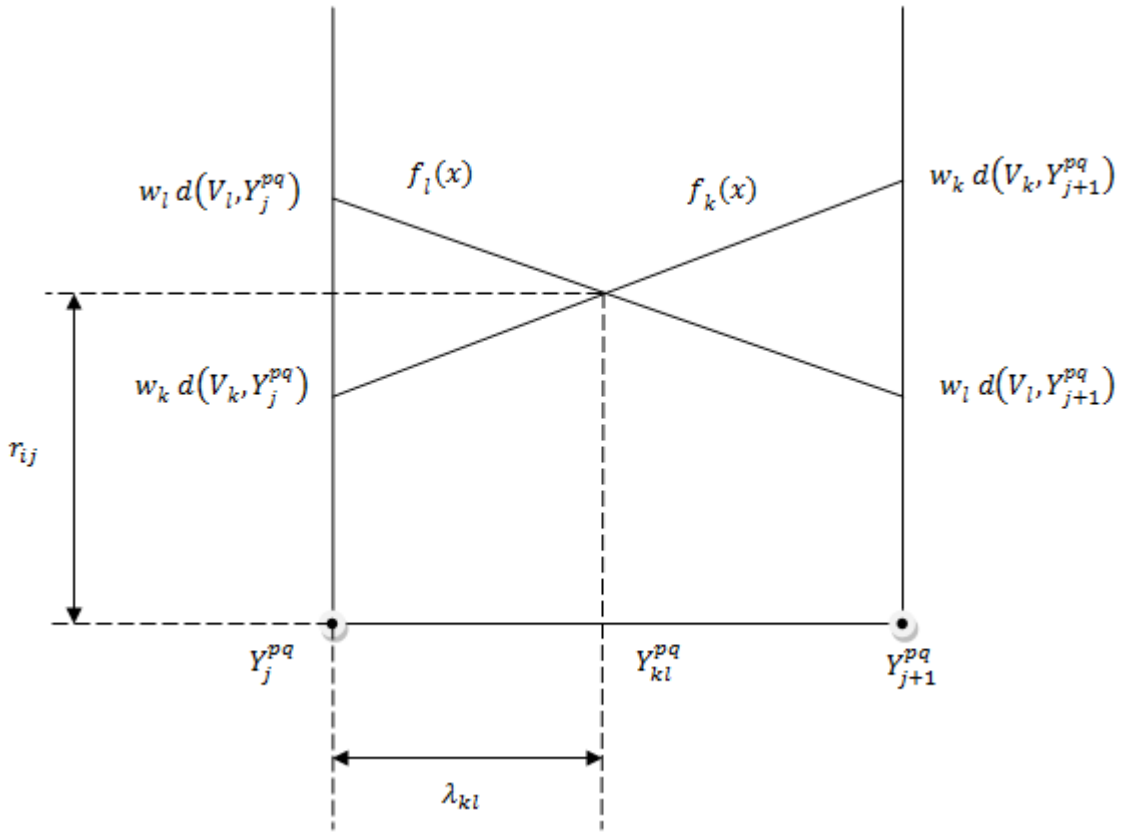


Figure 12 The illustration of an intersection point

Substituting $d(V_l, Y_j^{pq}) = d(V_l, Y_{j+1}^{pq}) + l$ and solving for λ_{kl}, we get

$$\lambda_{kl} = \frac{w_l\, d(V_l, Y_j^{pq}) - w_k\, d(V_k, Y_j^{pq})}{w_l + w_k}$$

Then Y_{kl}^{pq} is defined to be the point of intersection on $[Y_j^{pq}, Y_{j+1}^{pq}]$ with distance λ_{kl} from Y_j^{pq}.

The radius r_{kl} of the intersection point-radius pair (Y_{kl}^{pq}, r_{kl}) is computed as follows:

$$r_{kl} = w_k\, d(V_k, Y_j^{pq}) = w_l\, d(V_l, Y_j^{pq})$$

The measure of λ_{kl} and r_{kl} enables us to identify the location of the intersection point-radius pair on the specified edge segment and it's illustrated in Figure 12.

The intersection point-radius pairs of the example are calculated by checking the conditions 1 and 2 and are listed as follows:

	$V_k - V_l$	$[p,q]$	λ_{kl}^{pq}	r_{kl}^{pq}
1	V4-V2	[1,2]	0	3
2	V5-V3	[1,2]	0	6
3	V6-V3	[1,2]	1	5
4	V1-V2	[1,2]	1.5	1.5
5	V4-V3	[1,2]	1.5	4.5
6	V6-V5	[1,2]	2	6
7	V1-V3	[1,2]	3	3
8	V2-V4	[1,4]	0	3
9	V5-V3	[1,4]	0	6
10	V6-V3	[1,4]	1	5
11	V1-V4	[1,4]	1.5	1.5
12	V2-V3	[1,4]	1.5	4.5
13	V6-V5	[1,4]	2	6
14	V1-V3	[1,4]	3	3
15	V2-V6	[1,6]	0.5	3.5
16	V3-V5	[1,6]	0	6
17	V1-V3	[1,6]	4	4
18	V1-V5	[1,6]	3	3
19	V1-V6	[1,6]	2	2
20	V2-V3	[1,6]	2.5	5.5

	$V_k - V_l$	$[p,q]$	λ_{kl}^{pq}	r_{kl}^{pq}
21	V2-V3	[1,6]	2.5	5.5
22	V2-V5	[1,6]	1.5	4.5
23	V1-V3	[2,3]	0	3
24	V1-V5	[2,3]	1	4
25	V4-V5	[2,3]	0.5	4.5
26	V1-V4	[2,3]	1.5	4.5
27	V1-V6	[2,3]	2	5
28	V2-V3	[2,3]	1.5	1.5
29	V2-V4	[2,3]	3	3
30	V2-V5	[2,3]	2.5	2.5
31	V1-V4	[2,4]	0.5	3.5
32	V2-V4	[2,4]	2	2
33	V2-V1	[2,4]	3.5	3.5
34	V2-V1	[3,4]	1.5	4.5
35	V2-V4	[3,4]	0	3
36	V3-V4	[3,4]	1.5	1.5
37	V5-V1	[3,4]	2	4
38	V5-V4	[3,4]	0.5	2.5
39	V6-V1	[3,4]	1	5
40	V3-V1	[3,4]	3	3

	$V_k - V_l$	$[p,q]$	λ_{kl}^{pq}	r_{kl}^{pq}
41	V5-V2	[3,4]	2.5	4.5
42	V2-V6	[3,5]	0.5	3.5
43	V3-V5	[3,5]	1	1
44	V3-V6	[3,5]	2	2
45	V2-V1	[5,6]	0.5	5.5
46	V3-V1	[5,6]	2	4
47	V3-V6	[5,6]	0	2

The first column represents the pair of vertices V_k, V_l that induces the intersection point-radius pair. The second column gives the edge of the intersection point-radius pair whereas the distance to vertex V_p and the radius of the intersection point-radius pair are presented in the third and fourth columns, respectively. In addition to these intersection point-radius pairs, the vertices themselves are also considered as intersection point-radius pairs.

Evaluation of the Shortest Distance Matrix

After identifying the intersection point-radius pairs, shortest path distance matrix between each intersection point and each vertex is generated and tabulated as follows:

$$\begin{pmatrix} 0 & 3 & 6 & 3 & 6 & 4 \\ 0 & 3 & 6 & 3 & 6 & 4 \\ 1 & 2 & 5 & 4 & 7 & 5 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 6 & 5 & 2 & 5 & 0 & 2 \\ 5 & 6 & 3 & 6 & 1 & 1 \end{pmatrix}$$

The matrix above indicates the shortest path distances between 47 intersection points and 6 vertices. Next, the radius set of the network is generated. The radius set contains distinct radius values of intersection point-radius pairs and sorted in ascending order. The radius set of the example is as follows:

$$\text{Radius set} = \{0, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6\}$$

The generation of the shortest distance matrix and the radius set ends the preprocessing phase of the algorithm.

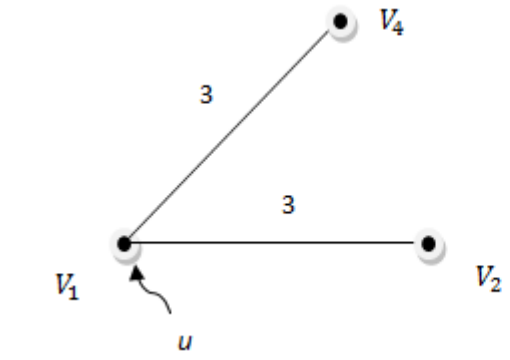
Construction of the Mathematical Model

By using the intersection point-radius pairs, subtrees rooted from the intersection point-radius pairs $X\left(\lambda_{ij}^{pq}, r_{ij}^{pq}\right)$ are constructed. The trees are constructed such that $T\left(x, r_{ij}^{pq}\right) = \left\{V_k \in V : w_k\, d(x, V_k) \leq r_{ij}^{pq}\right\}$. That is, the point x is taken to be the root and all vertices that are reachable from the root within the radius value r_{ij}^{pq} are included in the tree. Let $T_1, T_2, \ldots, T_l$ be an enumeration of these trees with radius $r_1, r_2, \ldots, r_l$ where l is the number of intersection point-radius pairs of the network. Associated with each tree T_j, let U_j be the root of the tree T_j where U_j is either an intersection point or a vertex in the original vertex set V.

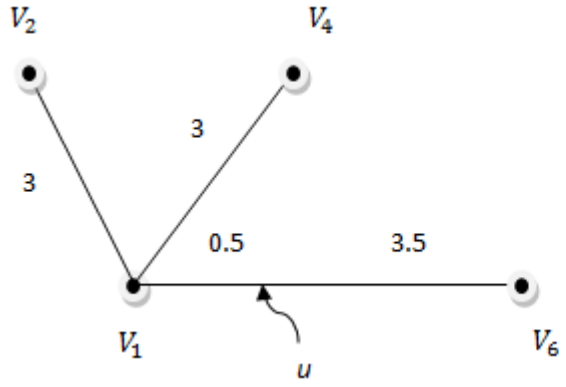
Consider the following three intersection point-radius pairs of the example.

$V_i - V_j$	$[p, q]$	λ_{ij}^{pq}	r_{ij}^{pq}
V4-V2	[1,2]	0	3
V2-V6	[1,6]	0.5	3.5
V6-V1	[3,4]	1	5

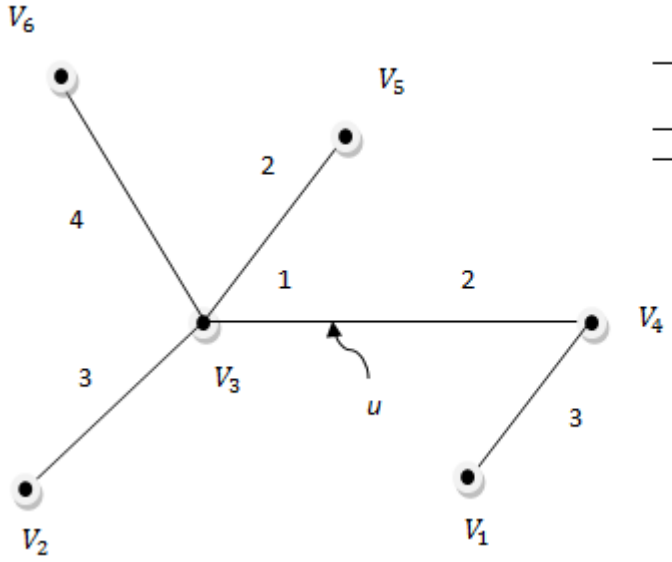
The constructed trees for these intersection point-radius pairs of the example are illustrated in Figure 13.



$V_i - V_j$	$[p,q]$	λ_{ij}^{pq}	r_{ij}^{pq}
V4-V2	[1,2]	0	3



$V_i - V_j$	$[p,q]$	λ_{ij}^{pq}	r_{ij}^{pq}
V2-V6	[1,6]	0.5	3.5



$V_i - V_j$	$[p,q]$	λ_{ij}^{pq}	r_{ij}^{pq}
V6-V1	[3,4]	1	5

Figure 13 The trees formed by intersection point-radius pairs of the example

Form the n by l matrix $A = [a_{ij}]$ as follows:

$$a_{ij} = \begin{cases} 1 & \textit{if vertex } V_i \textit{ is included in the tree } T_j \quad i.e \textit{ if } \; w_i\, d(V_i, U_j) \le r_j \\ 0 & \textit{otherwise} \end{cases}$$

Column a_j of matrix A is associated with the tree T_j and is formed as soon as we find an intersection point-radius pair that defines the tree T_j.

The matrix A of the example under consideration is as follows:

15th ↓ 22nd ↓ 39th ↓

0	0	1	1	0	0	1	0	0	0	0	0	1	1	1	1	1	1	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	0	1	0	0	1	0	0	1	0	1	1	1	0	0	1	1	1	0	1	1	1	1	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	0	1	1	0	0	1	1	1	1	0	1	0	1	0	1	1	1	1	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	0	1	0	1	0	0	0	1	1	0	1	1	0	0	1	1	1	1	1	1	1	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	0	0	0	0	1	1	1	1	0	0	0	0	1	0	1	1	0	0	0	0	1	1	1	1	1	0	0	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	0	1	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	1	0	0	1	0	0	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1

When we check the matrix A of the intersection points-radius pairs, the 15th column indicates that the tree T_{15} is composed of V_1, V_2 and V_4, the 22nd column indicates that the tree T_{22} is composed of V_1, V_2, V_4 and V_6, and the 39th column indicates that the tree T_{39} is composed of all vertices in the network. These trees are previously illustrated in Figure 13.

Finally, we solve a finite series of minimum set cover problems so that p of the trees cover the vertex set while the largest radius value associated with the selected set of trees is minimum.

Recall that at each set covering problem, the model finds an objective function indicating the number of centers to be located in the network. Moreover, the model performs a binary search on the ordered set of radius values of the network. The model seeks for the smallest radius value r in the finite set $\{r_1, \dots, r_m\}$ whose objective value is less than or equal to p.

Let $q(r)$ be the optimal objective function value of the set covering problem for a specific radius value r. If $q(r) > p$ then we need to increase the value of the radius r since more than p points are needed to cover all vertices within a radius value of r. We increase the radius value with the binary search on the radius set of the network. If $q(r) \leq p$ then we need to decrease the value of the radius r since this radius value r permits to place at most p centers on the network and still cover all vertices within a radius value of r. We stop with the smallest radius value r for which $q(r) \leq p$.

Suppose that we intend to locate two centers. Consider the ordered radius set of the example:

$$\text{Radius set} = \{0, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6\}$$

The binary search begins with the radius value at the middle of the radius set which is $r = 3.5$. For $r = 3.5$ the set cover problem locates two centers i.e the case $q(r) \leq p$.

Now the radius set that we are interested in becomes:

$$\text{Radius set} = \{0, 1, 1.5, 2, 2.5, 3, 3.5\}$$

We do the same binary search on the new radius set. The middle of this radius set gives $r = 2$. For $r = 2$ the set cover problem locates three centers i.e the case $q(r) > p$.

Now the radius set we are interested in becomes:

$$\text{Radius set} = \{\ 2.5, 3, 3.5\}$$

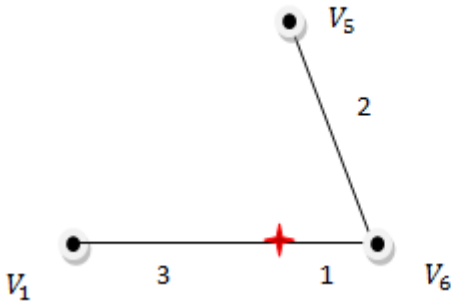
The middle of this radius set gives $r = 3$. For $r = 3$ the set cover problem locates two centers i.e the case $q(r) \leq p$.

Now the radius set we are interested in becomes:

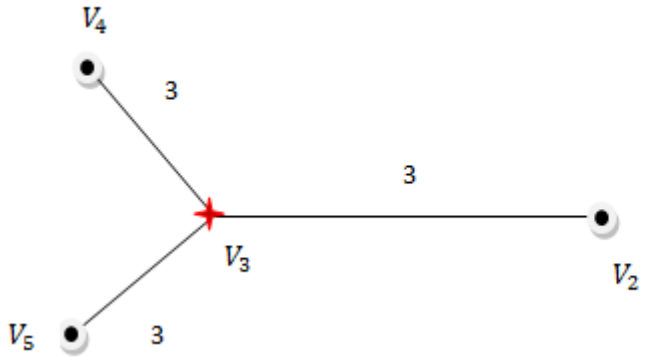
$$\text{Radius set} = \{2.5, 3\}$$

We pick $r = 2.5$ because at the previous iteration we picked the radius value $r = 3$. For $r = 2.5$, the set cover problem locates three centers. The new radius set now becomes $\{3\}$. Since $q(r = 3) = 2$, the algorithm terminates with two centers found as the solution of the set covering problem with $r = 3$. The trees of these centers and the locations on the network are illustrated in Figure 14 and Figure 15, respectively.

In Figure 14, the first tree has a radius value $r = 3$ and composed of vertices V_1, V_5 and V_6, whereas the second tree has a radius value $r = 3$ and composed of vertices V_2, V_3, V_4 and V_5. Note that the union of these trees contain all of the vertices in network N. In Figure 15, the first center is located on edge $[V_1\ V_6]$ with a three unit distance to V_1 and the second center is located on vertex V_3.



$V_i - V_j$	$[p,q]$	λ_{ij}^{pq}	r_{ij}^{pq}
V1-V5	[1,6]	3	3



$V_i - V_j$	$[p,q]$	λ_{ij}^{pq}	r_{ij}^{pq}
V2-V4	[2,3]	3	3

Figure 14 Illustration of the trees formed by the centers resulted from example

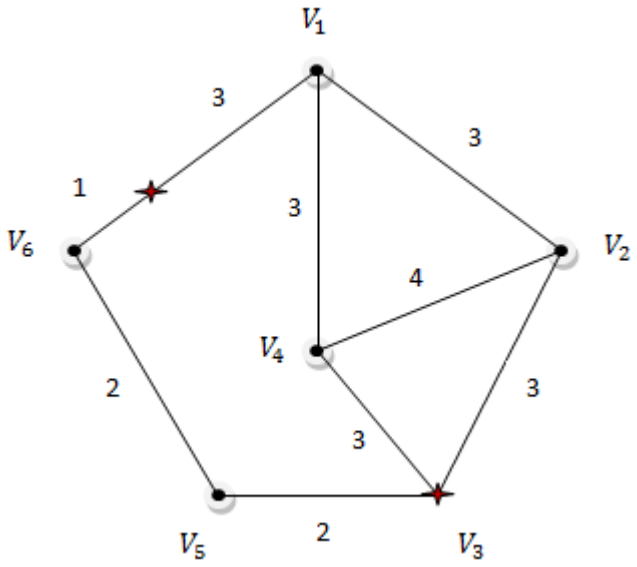


Figure 15 The location of the centers on network

3.5 Comparison Between Proposed and Classical Methods

There are significant differences between the proposed and classical method in terms of construction of the set covering matrices. In the classical method, a finite series of set covering problems is solved in order to obtain the p-center solution, but the construction of the set covering matrix is done differently.

Let $U_1, U_2 \ldots U_t$ be an enumeration of candidate points that are either vertices in V or intersection points. Let r be a fixed radius of coverage. Define

$$X_j = \begin{cases} 1 & if\ the\ candidate\ point\ U_j\ is\ selected\ as\ a\ center \\ 0 & otherwise \end{cases}$$

$$a_{ij} = \begin{cases} 1 & if \quad w_i\, d(V_i, U_j) \le r \\ 0 & otherwise \end{cases} \qquad i \in \{1,2, \ldots n\}, j \in \{1,2, \ldots t\}$$

$$\text{q(r)} = \min \sum_{j=1}^{t} X_j \qquad (1)$$

s.t

$$\sum_{j=1}^{t} a_{ij} X_j \ge 1 \qquad i \in I \qquad (2)$$

$$X_j \in \{0,1\} \qquad j \in \{1,2, \ldots . t\} \qquad (3)$$

Although, at first look, the model seems to be similar to the proposed method there are important differences. First, in the proposed method we consider the candidate points as intersection point-radius pairs and vertices whereas in the classical method the candidate points are the intersection points and the vertices. If an intersection point-radius pair induces

more than one such tree corresponding to different radius values, these trees are treated as distinct trees in the proposed method, but in the classical method only intersection points and vertices are taken into consideration as a unique point in the network. In order to be more clear, consider the first two intersection point-radius pairs of the example investigated previously.

$V_i - V_j$	$[p, q]$	λ_{ij}^{pq}	r_{ij}^{pq}
V4-V2	[1,2]	0	3
V5-V3	[1,2]	0	6

These two intersection point-radius pairs correspond to a single point located on vertex V_1 in the network. In the proposed method, we treat this point as two distinct trees with radius values 3 and 6 respectively. The point of view of the classical method is that regardless of its radius value there is a single point located on vertex V_1. These two different perspectives makes significant changes on the construction of the matrix $A = [a_{ij}]$.

At each set covering problem, the proposed algorithm restricts the number of candidate points to a set $J(r)$, where $J(r)$ is the set of j in J such that $r_j \leq r$. This index set ensures that the columns (trees) under consideration have radii no larger than the selected radius r. In the classical method, there is no restriction on the candidate points and all of the candidate points are taken into consideration in each attempted set covering problem. (Figure 16)

	Classical Method	**Proposed Method**
Construction of matrix A	$a_{ij} = \begin{cases} 1 & if \ \ w_i\, d(V_i, U_j) \le r \\ 0 & otherwise \end{cases}$	$a_{ij} = \begin{cases} 1 & if \ \ w_i\, d(V_i, U_j) \le r_j \\ 0 & otherwise \end{cases}$
Objective Value	$\min \sum_{j=1}^{t} X_j$	$\min \sum_{j \in J(r)} X_j$
Constraint	$\sum_{j=1}^{t} a_{ij} X_j \ge 1 \qquad i \in I$	$\sum_{j \in J(r)} a_{ij} X_j \ge 1 \qquad i \in I$
Binary Variable	$X_j \in \{0,1\} \quad j \in \{1,2,\ldots . t\}$	$X_j \in \{0,1\} \qquad j \in J(r)$

Figure 16 Comparison between classical and proposed method

As mentioned before, there are substantial differences during the construction of the matrix A. Consider the following intersection point-radius pairs.

	$V_i - V_j$	$[p, q]$	λ_{ij}^{pq}	r_{ij}^{pq}
1	V4-V2	[1,2]	0.5	3
2	V5-V3	[1,2]	0.5	6.5
3	V6-V3	[1,2]	1	2
4	V1-V2	[1,2]	1	8
5	V4-V3	[1,2]	2	3.5
6	V6-V5	[1,2]	2	7
7	V1-V3	[1,2]	2.5	1

The matrix A of these intersection point-radius pairs for the proposed method is as follows:

	r = 3, 6.5 ↓ $X(0.5)$	r = 2, 8 ↓ $X(1)$	r = 3.5, 7 ↓ $X(2)$	r = 1 ↓ $X(2.5)$
V_1	1	1	1	0
V_2	1	1	1	1
V_3	1	1	1	0
V_4	1	1	1	0
V_5	1	1	1	0
V_6	1	1	1	0

Figure 17 The matrix A of the proposed method

Note that the rows and columns correspond to vertices and intersection point-radius pairs, respectively. The A matrix of these candidate points for the classical method is as follows:

	$X(0.5)$	$X(1)$	X(2)	$X(2.5)$
V_1	1	1	1	0
V_2	0	1	1	1
V_3	0	0	0	0
V_4	0	0	0	0
V_5	0	0	0	0
V_6	0	0	0	0

Figure 18 The matrix A of the classical method

Suppose that after the binary search on the radius set the radius value results to be $r = 2$. In the proposed method, for $r = 2$ we take the highlighted columns of matrix A where $r_j \leq 2$ in order to solve the set covering problem.(Figure 19)

	r = 3, 6.5 ↓ $X(0.5)$	r = 2, 8 ↓ $X(1)$	r = 3.5, 7 ↓ $X(2)$	r = 1 ↓ $X(2.5)$
V_1	1	1	1	0
V_2	1	1	1	1
V_3	1	1	1	0
V_4	1	1	1	0
V_5	1	1	1	0
V_6	1	1	1	0

Figure 19 Selected columns of matrix A in the example

In the classical method, for $r = 2$ we take all the columns of matrix A (Figure 18) in order to solve the set covering problem.

Although this is a representation of a small-sized problem, we eliminate two columns by using the proposed method. When we significantly increase the size of a network, the number of eliminated columns in the proposed method becomes worthwile. When we have significant number of columns to eliminate, the advantage of the proposed method with respect to the classical one arises in terms of cpu times, number of branch and bound nodes, and the number of iterations used to solve the algorithm.

During the set covering problems, the classical method generates matrix A at each set covering problem according to the radius value *r* selected during the binary search on the candidate radius set. In the proposed method, regardless of how many set covering problems are required to solve the p-center problem, we generate the matrix A only once then delete its columns as necessary depending on *r*. The difference comes from the fact that we consider the candidate center locations with a corresponding radius value r_j whereas, in the classical method, the candidate center locations are only points without any radius value. The illustration in Figure 20 indicates that whenever we calculate the intersection-point radius pairs in the proposed method, we form the matrix A, but in the classical method at each set covering problem and at each binary search matrix A is calculated over and over.

	Proposed Method	**Classical Method**
Construction of matrix A	$a_{ij} = \begin{cases} 1 & if \quad w_i d(V_i, U_j) \leq r_j \\ 0 & otherwise \end{cases}$	$a_{ij} = \begin{cases} 1 & if \quad w_i d(V_i, U_j) \leq r \\ 0 & otherwise \end{cases}$

Figure 20 Construction of matrix A in proposed and classical method

Instead of forming matrix A over and over, after the calculation of matrix A, we search for the columns such that $r_j \leq r$ at each set covering problem of the proposed approach. The differences between proposed and classical method are summarized in Figure 21.

		Proposed Method	Classical Method
Preprocessing Phase	**Step 1**	Calculate intersection point-radius pairs via antipodals.	Calculate intersection points.
	Step 2	Form shortest path distance matrix between each intersection point and vertices.	Form shortest path distance matrix between each intersection point and vertices.
	Step 3	Form radius set by the distinct radius values of the intersection point-radius pairs. Enumerate radius set in ascending order.	Form radius set by the distinct values of the shortest path distances between the vertices and intersection points. Enumerate radius set in ascending order.
Mathematical Model Phase	**Step 4**	Form matrix A by: $a_{ij} = \begin{cases} 1 & if \quad w_i d(V_i, U_j) \le r_j \\ 0 & otherwise \end{cases}$ where $d(V_i, U_j) \le r_j$ is the shortest path distance between each intersection point-radius pair and vertex.	Make binary search on radius set. Let the radius value results to be r.
	Step 5	Do a binary search on the radius set. Let the selected radius value be r.	Form matrix A by: $a_{ij} = \begin{cases} 1 & if \quad w_i d(V_i, U_j) \le r \\ 0 & otherwise \end{cases}$ where $d(V_i, U_j) \le r$ is the shortest path distance between each intersection point and vertex.
	Step 6	Select the corresponding intersection point-radius pairs for which $r_j \le r$ and solve the set covering problem by using matrix A formed by the intersection point-radius pairs such that $r_j \le r$	Solve the set covering problem by using the matrix A formed for all intersection points calculated in preprocessing phase.
	Step 7	If $q(r) > p$ or $q(r) \le p$ go back to step 5.	If $q(r) > p$ or $q(r) \le p$ go back to step 4.
	Step 8	Solve a finite series of set covering problems until at most p number of centers are located with minimum radius value.	Solve finite a series of set covering problems until at most p number of centers are located with minimum radius value.

Figure 21 The comparison between proposed and classical method algorithm

Chapter 4

IMPLEMENTATION

In this thesis one of our purposes is to perform computational analysis on the proposed algorithm and compare it with the classical approach in the literature in terms of computational efficiency. During this comparison, we solve medium and large scale test problems that were previously generated. In addition to these previously generated test problems, we developed a Turkish road network in order to test the algorithm in a planar network. Throughout this chapter implementation of the algorithm is described in detail.

4.1 Implementation of the Algorithm

Despite the fact that our main concern in this study is the computational analysis and comparison, we initially concentrate on choosing the most appropriate programming language for the proposed and classical algorithms. We determine to use C as the programming language. C programming language has many strengths. It is flexible and portable, it can produce fast and compact code, it provides the programmer with objects to create and manipulate complex structures and provides low level routines to control hardware.

The reasons for choosing C programming language are stated as follows:

- **Availability:** C language is almost certainly available on more machines than any other programming language.

- **Portability:** C programs can be written in such a way that the programs can easily be transferred from one C compiler to another. C compiler has an advantage to port programs to new compiler or new machine due to the fact that C programming language does not specify a standard library routine.

- **Efficiency:** The term efficiency has many meanings. It could mean any of the following:
  - ✓ Ability to produce programs which run quickly.
  - ✓ Production of small executable programs.
  - ✓ Being able to write a given algorithm in fewer program statements.

  All the above features are available in C language.

- **Modular Programming and Libraries:** C provides the ability to split large programming tasks into separate modules. Each module can consist of a C source file which can be compiled separately from all the other modules. With careful use of static external variables, a certain amount of information hiding can be performed.

- **Applications:** Virtually all new operating systems are written in C programming language. C is also used for many commercial programs where speed of program execution or the time taken to produce the program is of vital importance.

The algorithm is implemented via the commercial product Microsoft Visual Studio 2008 and the mathematical model is solved by using CPLEX 12.1.

During the implementation of the algorithm there are various important points that are taken into consideration. The most significant feature is the space restriction where the memory must be used efficiently. In order to implement the algorithm efficiently, the most adequate data types and data structures must be used. We use the following data types and data structures during the implementation of the algorithm:

1) The data types that we use are *int, double, float, unsigned long* and *char* .

2) We use *arrays* in order to store related data under a single variable name with an index, also known as a subscript. It is easiest to think of an array as simply a list or ordered grouping of variables. As such, arrays often provide to organize collections of data efficiently and intuitively in terms of memory requirement.

3) We use *linked list* in order to provide an easy implementation for several important abstract data structures. The principal benefit of a linked list over a conventional array is that the order of the linked items may be different from the order that the data items are stored in memory or on disk. For that reason, linked lists allow insertion and removal of elements at any point in the list, with a constant number of operations.

The implementation of the algorithm consists of two interdependent C codes. The first code evaluates the intersection point-radius pairs whereas the second code solves finite series of set covering problems. The codes are interdependent because of the fact that the output of the first code is the input for the second code.

In order to ensure clarity of the procedure; implementation of the algorithm is stated step by step as follows:

1) The user defines the following parameters to initialize the first code:

- ✓ Number of Vertices
- ✓ Number of Demand Vertices
- ✓ Weights of Vertices

By defining parameters we mean the size of the network i.e the number of vertices and the number of demand vertices. For most cases, the two are the same.

2) The first code for the preprocessing stage of the algorithm requires generating two input files. The first input file consists a square matrix of the shortest path distances between each pair of vertices of the network whereas the second input file is composed of the edge lengths of the network.

3) By using the data in the input files we first calculate antipodals; then we evaluate intersection point-radius pairs via antipodals.

4) The output results consist of a matrix indicating the shortest path distances between each intersection point-radius pair and vertex. In addition we obtain the radius set of the corresponding intersection point-radius pairs.

5) The output results of the preprocessing phase are integrated to the second code as input files.

6) The user defines the following parameters to initialize the second code:

- ✓ Number of intersection point-radius pairs.
- ✓ Number of intersection points.
- ✓ Number of radii in the radius set.
- ✓ Number of centers to be located.

7) The matrix indicating the shortest path distances between each intersection point-radius pair and each vertex is our distance matrix between candidate center locations and vertices. By using this matrix we form the matrix A consisting of zero's and one's.

8) By using the radius set obtained from the preprocessing, we do a binary search and solve the set covering problem. After each binary search, we choose a new radius value and by using these radius values we solve a finite series of minimum set covering problems.

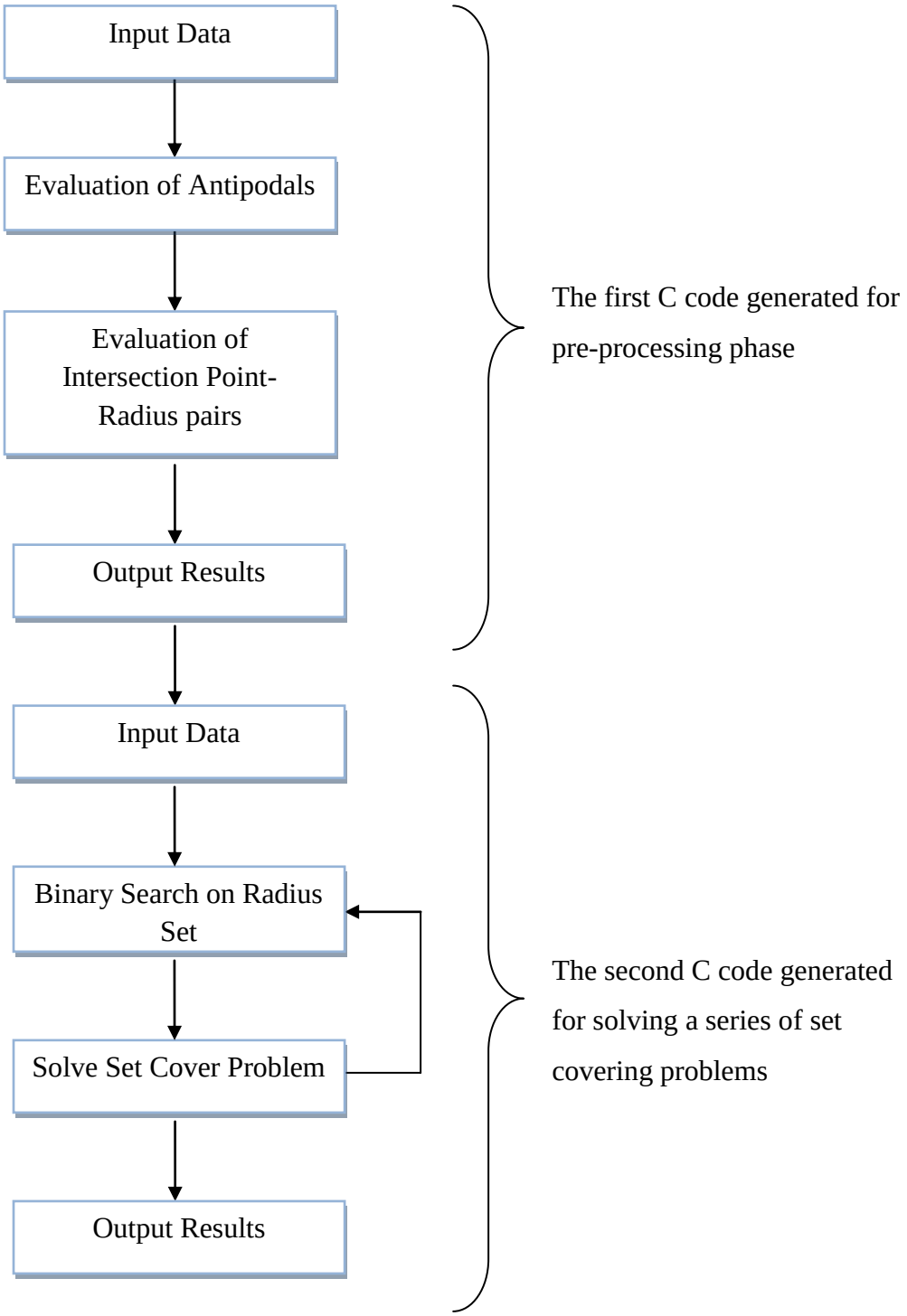
9) The binary search ends when we obtain the minimum radius value for locating p-centers. Note that in order to do a binary search on a radius set we organize the radius values of the intersection points in ascending order.

10) The output results are composed of the following parameters:

1) Optimal radius value.
2) Number of branch and bound nodes.
3) CPLEX time.
4) The location of the selected centers.
5) Number of variables.
6) Number of iterations.
7) LP Bounds.

We clarify the output parameters in the forthcoming chapter of the report.

The pseudo-code for our algorithm is stated as follows:



Chapter 5

COMPUTATIONAL ANALYSIS

5.1 Test Problems

In our computations, we consider absolute and vertex restricted p-center problems for unweigthed cases. The computations are performed on planar and non-planar networks. For planar networks, we use real data from the Turkish road network and for non-planar networks, we use problem instances from the OR library (Beasley, 1985).

The size of the test problems from the OR library varies from 100 vertices to 900 vertices whereas the Turkish road network has 343 vertices. Number of edges and number of intersection points are the other parameters indicating the size of the network. The sizes of the test problems from the OR library and Turkish road network and the number of intersection points generated are given in Table 1.

# of Vertices	# of DV	# of Edges	# of Int. Pts.
100	100	400	19488
200	200	1600	66059
300	300	3600	135320
400	400	6400	204551
500	500	10000	285055
600	600	14400	381157
700	700	19200	458850
800	600	25100	518814
900	600	31800	607583
343	343	1004	49257
343	81	1004	25292

Table 1 The size of the test problems

An instance of the problem is defined by the following factors: number of vertices, number of demand vertices, number of intersection points and number of centers of the network. For each combination of these factors several number of instances are generated. Note that the term *demand vertex* corresponds to the vertices that are required to be covered.

5.2 Performance Measures

During the computational analysis of the integer programming based approaches, the following performance measures are taken into consideration:

1) Number of branch and bound nodes.
2) CPU time.
3) Number of variables.
4) Number of iterations.
5) LP bounds.

Branch and bound nodes enable to access the number of nodes used and CPU time is the time required to solve the integer program. We measure number of variables by summing up the number of intersection points and number of vertices of a network. Iteration number indicates the number of iterations that CPLEX uses in order to optimize the integer programming model. We indicate and measure LP bounds as currently the best known bound on the optimal solution value of the integer problem.

The computational analysis not only includes the performance measure of the proposed algorithm but also gives a comparison to the classical method in the literature. For this purpose, we developed codes for both algorithms. The preprocessing phase and the integer programming for the mathematical model are both coded in C programming language. Integer programming models are solved by using CPLEX 12.1 and run on a personal computer with a 2.40 GHz Intel Core 2 Duo processor and 6 GB of RAM.

5.3 Developing the Turkish Road Network

To comprehend the structure of the Turkish road network, we interviewed General Directorate of Highways (KGM) and Greater Municipality of Ankara (ABB). During the interviews, we obtained the Turkish road network from General Directorate of Highways in electronic format. The Turkish road network is studied and surveyed according to the data obtained from satellite images. We verified the Turkish road network in terms of up-to-dateness and accuracy by investigating the most recent printed version road network of Turkey. During our investigations, we made slight modifications such as inserting several roads that have been recently developed or deleting couple of roads that are no longer in use.

During the development of the Turkish road network, we considered highways, two-lane roads, three-lane-roads and four-lane-roads. Rural roads are not taken into consideration because our attention is on the road network that includes cities and country boroughs.

Another reason is that although rural roads may give advantage in terms of actual lengths, the travel time would be much longer in comparison to other high quality roads.

Our Turkish road network consists of 1004 edges and 343 vertices. The vertices are designated in accordance with the terminal points of the arcs. The lengths of each edge are recorded according to the most recent printed version of the Turkish road network and the corresponding names of each vertex in Turkish network are placed on record. Lastly, the neighbors of each vertex are identified in order to develop a shortest distance matrix between each vertex. Note that two vertices are neighbors if they are directly connected by an arc.

5.4 Computational Results for the Absolute P-Center Problem

As mentioned previously, we performed a series of computational experiments for the absolute p-center problem. The computational results for the instances obtained from OR library and currently developed Turkish road network is illustrated in Table 2.

					Proposed Method				Classical Method			
n	**p**	**Opt rad**	**Edge #**	**Var #**	**Iter**	**NBB**	**Gap%**	**Cpu**	**Iter**	**NBB**	**Gap%**	**Cpu**
100	1	185	400	19488	1081	0	0	39.2	874	0	0	108
100	3	140	400	19488	2145	18	0	35.2	1724	13	0	98.6
100	5	115.5	400	19488	1836	0	0	26.7	1791	10	0	94.2
100	10	88	400	19488	4703	86	0	26.7	3726	90	0	94.5
100	15	70.5	400	19488	1752	0	0	22.8	1969	0	0	101.1
100	20	58.5	400	19488	2070	0	0	26.8	1978	0	0	92.4
100	25	50.5	400	19488	2018	0	0	25	1996	0	0	93.3
200	1	108	1600	71616	774	0	0	401.8	2784	11	0	406.3
200	3	90.5	1600	71616	4216	34	0	323.4	2851	11	0	345.7
200	5	79.5	1600	71616	4586	34	0	305.5	3594	11	0	326.5
200	7	73	1600	71616	12915	1244	0	314.8	16588	1929	0	314.2
200	15	53	1600	71616	17952	780	0	269.8	16570	877	0	295
200	20	44.5	1600	71616	10041	98	0	280.1	9916	137	0	286.6
200	25	39.5	1600	71616	8832	78	0	298.8	8384	32	0	301
300	5	56	3600	135320	1911	3	0	783.3	3075	2	0	742
300	10	46.5	3600	135320	4248	16	0	728.2	4707	15	0	687.3
300	15	41.5	3600	135320	22924	788	0	689.4	22924	988	0	689.4
300	24	34.5	3600	135320	14231	710	0	618.4	12199	585	0	584.2
300	30	31.5	3600	135320	6398	0	0	667.7	6957	52	0	621.7
400	3	47	6400	204551	1007	0	0	1528.7	1134	0	0	1457.3
400	5	44.5	6400	204551	1687	0	0	1515	1236	0	0	1439.9
400	12	36	6400	204551	4237	53	0	1428.4	4518	53	0	1348.9
400	20	30	6400	204551	8464	36	0	1360.3	7749	19	0	1249.9
400	25	27.5	6400	204551	8484	92	0	1353.6	8942	103	0	1257.4
400	30	25.5	6400	204551	10632	39	0	1424.6	8773	24	0	1264.4
500	10	31.5	10000	285055	7336	118	0	1979.6	15814	63	0	1735.2
500	15	28	10000	285055	10509	36	0	1712.5	19374	492	0	1605.7
500	25	24	10000	285055	9249	35	0	1718.5	22685	446	0	1547.7
500	35	21.5	10000	285055	16752	231	0	1858.5	19272	321	0	1531.1
600	26	23.5	14400	381757	11411	120	0	4318.2	16863	220	0	4177.4
600	33	21.5	14400	381757	14064	79	0	4094.5	18514	213	0	4181.9
600	44	19.5	14400	381757	20520	43	0	3968.9	23446	174	0	4132.3
600	78	14.5	14400	381757	9013	8	0	3144.4	15220	36	0	4078.9
700	11	25	19202	458850	11918	132	0	4567.4	15337	462	0	4968
700	24	20	19202	458850	45570	943	0	4900	49850	1023	0	5437.6
700	43	16.5	19202	458850	96457	1095	0	5928	75750	1153	0	6962
700	74	13	19202	458850	21988	179	0	4541.3	27116	179	0	5805
700	93	11.5	19202	458850	23116	171	0	4981.6	32210	296	0	6448.6
800	35	16	25098	518814	20466	278	0	4301.2	19097	282	0	5081.6
800	44	14.5	25098	518814	26814	288	0	4698.1	25291	273	0	5635.1
800	62	12.5	25098	518814	35831	468	0	4024.6	29487	479	0	4529.9
800	76	11	25098	518814	26750	249	0	4488.5	26306	256	0	5063.7
800	100	9.5	25098	518814	33440	352	0	4507.9	33166	355	0	5282.4
900	28	15.5	31796	607583	22662	355	0	4351.2	29008	496	0	5014.6
900	39	14	31796	607583	21515	218	0	4735.7	27046	392	0	5562.7
900	49	12.5	31796	607583	17264	106	0	4172.2	26439	324	0	4890.3
900	60	11.5	31796	607583	23180	113	0	5293.8	25577	208	0	6087.2
900	84	9.5	31796	607583	15093	38	0	4634.5	20878	155	0	5535.1
TN(343DV)	1	1140	1004	49257	431	0	0	539.9	134	0	0	432.3
TN(343DV)	3	570	1004	49257	4408	0	0	468.5	1534	0	0	375.6
TN(343DV)	5	477	1004	49257	9246	0	0	411.4	1993	0	0	340.1
TN(343DV)	10	314	1004	49257	25744	1305	0	411.8	20404	3421	0	349.8
TN(343DV)	15	247	1004	49257	15008	147	0	366.1	7004	158	0	521.4
TN(81DV)	1	1065	1004	25292	484	0	0	47.2	98	0	0	48.5
TN(81DV)	3	542	1004	25292	1740	0	0	46.7	735	0	0	42
TN(81DV)	5	417.5	1004	25292	2382	0	0	45	2211	14	0	40.4
TN(81DV)	10	267	1004	25292	3180	10	0	48	2865	43	0	40.4
TN(81DV)	15	191	1004	25292	2741	9	0	44.4	2753	14	0	40.5
Average	-	-	-		13069	192	0	1781	13883	287	0	1971
Maximum	-	-	31796	607583	96457	1305	0	5928	75750	3421	0	6962

Table 2 Computational Results for Absolute p-center Problem

In Table 2, number of vertices of the network is illustrated in the 1st column. The number of centers and their corresponding optimal radii are listed in columns 2 and 3 respectively. The 4th column presents the number of edges whereas the 5th column points out the number of variables of the network. The number of variables of a network is identified by summing up number of intersection points and number of vertices. The columns under proposed and classical methods represent the iteration number, number of branch and bound node, percent gap and CPU time requirement in seconds respectively. We measure percent gap as the currently best known LP bound on the optimal solution value of the integer problem.

The observations in the computational analysis and comparison on absolute p-center problem are as follows:

- Out of 58 instances, the proposed method solved the algorithm with less number of branch and bound nodes than the classical method in 47 instances. The decrease in the number of branch and bound nodes indicates that the model initializes with a better solution and makes less branching.

- When we compare the results we observe that, on the average, the proposed method solves the algorithm with 192 branch and bound nodes whereas the classical method solves with 287 branch and bound nodes. This corresponds to an approximately 33 percent difference on the average number of branch and bound nodes.

- The difference between the two methods increases when we compare the maximum values obtained in terms of the number of branch and bound nodes. In the instance where p=10 in the Turkish network with 343 demand vertices, the proposed method solves the model with 1305 branch and bound nodes whereas the classical method solves with 3421 branch and bound nodes.

- CPLEX solved the instances in an average of 1781 seconds for the proposed method whereas the CPU time requirement increased to 1971 seconds in the classical method. Hence, the proposed method performs better in terms of the average CPU time requirement.

- The maximum CPU time requirement for the proposed method is around 5928 seconds while the maximum CPU time requirement for the classical method is around 6962 seconds. Thus, the proposed method performs better in terms of the maximum CPU time requirement as well.

- When we compare the number of iterations required to solve the problem, the proposed method solves with 13069 iterations on the average and the classical method solves with 13883 iterations on the average. This parameter indicates that the proposed method solves the same instances by using less effort.

- The gap percentage for both the proposed and the classical method is measured to be zero in all of the instances.

5.5 Computational Comparison between Absolute and Vertex-Restricted Networks

The vertex-restricted case of the p-center problem is suboptimal to the absolute case of the p-center problem. As mentioned before, the computational analysis in the literature is based on the vertex-restricted case of the p-center problem. In order to observe the differences between vertex-restricted and absolute cases in terms of optimality, we performed a computational analysis for the vertex-restricted case. In order to make comparison between the vertex-restricted and absolute cases of the p-center problem, we used the same instances that were performed for the absolute p-center problem. In order to solve the vertex-restricted p-center problem we use the same procedure as in the absolute case except we pick the vertices as the roots of the subtrees and form the shortest path trees via Dijkastra's algorithm that connects the root to the vertices reached within the radii of the vertices. The computational results and the comparison between vertex-restricted and absolute cases are illustrated in Table 3.

n	p	Abs. Opt rad	V.R Opt Rad	Gap %
100	1	185	186	0.54
100	3	140	148	5.41
100	5	115.5	121	4.55
100	10	88	91	3.30
100	15	70.5	79	10.76
100	20	58.5	68	13.97
100	25	50.5	62	18.55
200	1	108	112	3.57
200	3	90.5	93	2.69
200	5	79.5	82	3.05
200	7	73	76	3.95
200	15	53	56	5.36
200	20	44.5	49	9.18
200	25	39.5	45	12.22
300	5	56	57	1.75
300	10	46.5	49	5.10
300	15	41.5	43	3.49
300	24	34.5	36	4.17
300	30	31.5	33	4.55
400	3	47	48	2.08
400	5	44.5	45	1.11
400	12	36	37	2.70
400	20	30	32	6.25
400	25	27.5	29	5.17
400	30	25.5	27	5.56
500	10	31.5	34	7.35
500	15	28	30	6.67
500	25	24	26	7.69
500	35	21.5	24	10.42
600	26	23.5	25	6.00
600	33	21.5	23	6.52
600	44	19.5	21	7.14
600	78	14.5	16	3.00
700	11	25	25	0.00
700	24	20	21.5	6.98
700	43	16.5	17	2.94
700	74	13	14	7.14
700	93	11.5	13	11.54
800	35	16	17	5.88
800	44	14.5	15	3.33
800	62	12.5	13	3.85
800	76	11	12	8.33
800	100	9.5	10	5.00
900	28	15.5	16	3.13
900	39	14	14.5	3.45
900	49	12.5	13	3.85
900	60	11.5	12	4.17
900	84	9.5	10	5.00
TN(343DV)	1	1140	1141	0.09
TN(343DV)	3	570	580	1.72
TN(343DV)	5	477	481	0.83
TN(343DV)	10	314	331	5.14
TN(343DV)	15	247	260	5.00
TN(81DV)	1	1065	1081	1.48
TN(81DV)	3	542	560	3.21
TN(81DV)	5	417.5	426	2.00
TN(81DV)	10	267	279	4.30
TN(81DV)	15	191	211	9.48
Average	-	-	-	5.20
Maximum	-	-	-	18.55

Table 3 Comparison between Vertex-Restricted and Absolute Cases

In Table 3, the number of vertices and the number of centers are given in the 1st and 2nd columns, respectively. The optimal radius for absolute and vertex-restricted cases are listed in columns 3 and 4, respectively. The last column presents the percent difference between the absolute and the vertex-restricted radius value.

The observations for the comparison on absolute and vertex-restricted cases are as follows:

- We observe that on the average the absolute case of the problem is 5.20 percent better than the vertex-restricted case in terms of the optimal radius.

- The maximum difference between the absolute radius value and the vertex-restricted radius value is measured to be 18.55 percent.

- These differences in terms of optimally locating a facility are vital since the main application area of p-center problem is emergency service locations and the main concern is saving human life.

- Another observation is that in most of the instances, when the number of facilities(p) increases, the gap between optimal radius of the absolute and the vertex-restricted cases increases.

5.6 Computational Comparison between Complete and Incomplete Networks

In this section we give a computational comparison between complete and incomplete networks. By complete networks we mean that there is link between all vertices of the network whereas incomplete networks have links between certain pairs of vertices. The main reason for this comparison is to see how much additional computational effort is needed when an incomplete network problem is solved as a complete network problem. Converting incomplete network problems into complete network problems is traditionally what has been done in the literature by most researchers in solving various types of location problems. We want to see the adverse effects of this conversion (from incomplete to complete) in the context of the absolute p-center problem.

In order to do a computational comparison between complete and incomplete networks, we convert the previously used incomplete network with 100 vertices to a complete network. The conversion is made by considering the shortest path distances between all vertices as the edge lengths of the network. The computational results for incomplete and complete networks are tabulated in Table 4 and Table 5, respectively.

Incomplete Network												
					Proposed Method				Classical Method			
n	p	Opt rad	Edge #	Var #	Iter	NBB	Gap	Cpu	Iter	NBB	Gap	Cpu
100	5	115.5	400	19488	1836	0	0	26.7	1791	10	0	94.2
100	10	88	400	19488	4703	86	0	26.7	3726	90	0	94.5
100	15	70.5	400	19488	1752	0	0	22.8	1969	0	0	101.1
100	20	58.5	400	19488	2070	0	0	26.8	1978	0	0	92.4
100	25	50.5	400	19488	2018	0	0	25	1996	0	0	93.3
Average								25.6				95.1

Table 4 Computational Results for Incomplete Network

Complete Network												
					Proposed Method				Classical Method			
n	p	Opt rad	Edge #	Var #	Iter	NBB	Gap	Cpu	Iter	NBB	Gap	Cpu
100	5	115.5	10000	1234935	4471	0	0	2697.8	2712	0	0	2165.4
100	10	88	10000	1234935	7286	26	0	2506.2	11712	149	0	2105.9
100	15	70.5	10000	1234935	4202	0	0	2558.5	3471	0	0	2264.3
100	20	58.5	10000	1234935	4020	0	0	2488.2	3589	0	0	2162
100	25	50.5	10000	1234935	4349	0	0	2623.85	2918	0	0	2018.5
Average								2574.9				2143.2

Table 5 Computational Results for Complete Network

In Table 4 and Table 5, the number of vertices of the network is illustrated in the 1st column. The number of centers and their corresponding optimal radii are listed in columns 2 and 3, respectively. The 4th column presents the edge numbers whereas the 5th column points out the number of variables of the network. The columns under proposed and classical methods represent the iteration number, number of branch and bound node, gap percentage and CPU time requirement in seconds, respectively.

The observations in the computational analysis and comparison between complete and incomplete networks are as follows:

- The optimal radius is identical for both complete and incomplete networks. Hence, there is no difference in terms of optimality when we convert an incomplete network to a complete network.

- There are 400 edges for the incomplete network under consideration whereas the corresponding complete network has 10000 edges. Due to this difference in the edge numbers, the complete network instance has many more intersection points than the incomplete network instance. In the complete network, there

are approximately 1.2 million intersection points whereas in the incomplete version there are only 19488 intersection points.

- CPLEX solved the instances in an average of 2575 seconds for the proposed method in the complete network whereas the proposed method of the incomplete network CPU requirement is around 25 seconds. Hence, the complete network in the proposed method requires far more CPU time than the incomplete network.

- CPLEX solved the instances in an average of 2143 seconds for the classical method in the complete network whereas the classical method of the incomplete network CPU requirement is around 95 seconds. Hence, the complete network in classical method requires far more CPU time than the incomplete network.

- According to Table 4 and Table 5, there is no significant difference between the complete and incomplete networks in terms of number of branch and bound nodes.

- The gap percentage for both complete and incomplete networks is measured to be zero in all of the instances.

In order to do a computational comparison between complete and planar networks, we convert the previously used Turkish road network to a complete network. The conversion is made again by considering the shortest path distances between all vertices as the edge lengths of the network. The computational comparison between planar and complete version of the Turkish road network is tabulated in Table 6.

			Planar Network			Complete Network		
n	**p**	**Opt rad**	**Edge #**	**Var #**	**Cpu**	**Edge #**	**Var #**	**Cpu**
TN(81DV)	1	1065	1004	25292	47.2	117649	> 20.000.000	> 12 hrs.
TN(81DV)	3	542	1004	25292	46.7	117649	> 20.000.000	> 12 hrs.
TN(81DV)	5	417.5	1004	25292	45	117649	> 20.000.000	> 12 hrs.
TN(81DV)	10	267	1004	25292	48	117649	> 20.000.000	> 12 hrs.
TN(81DV)	15	191	1004	25292	44.4	117649	> 20.000.000	> 12 hrs.

Table 6 Computational Comparison between Planar and Complete Network

The observations in the computational analysis and comparison between planar and complete network for the proposed method are as follows:

- The number of edges for the planar network is around 1000 whereas the complete version of the network has more than 110000 edges.

- Due to the fact that the complete network has approximately hundred times more edges than the planar network, the number of variables of the complete network is excessively larger than the planar version of the network. The planar network has 25292 variables whereas the complete version has more than 20 million variables.

- The CPU time requirement of the planar network is around 45 seconds. While solving the complete version of Turkish road network, we limited the CPU time to 12 hours on CPLEX and optimization was not over within 12 hours. Hence, the complete version of Turkish road network has a CPU time requirement of more than 12 hours.

According to the results obtained from the tables of this section that although there is no difference in terms of optimality, there is a huge difference in terms of memory requirement and variable numbers between complete networks and planar or incomplete networks.

5.7 Investigation of LP Relaxation Bounds

In order to investigate the LP relaxation bounds for the IP formulations of the proposed and the classical methods, we replaced the binary constraints with non-negativity constraints and performed a binary search over the list of radii until the smallest radius is found for which the number of centers in the LP relaxation of the set covering solution is at most p. This yields a difference both in the optimal p-radius values of the IP and its LP relaxation and in the number of centers corresponding to that radius. The computational results and comparison are listed in Table 7.

In Table 7, the number of vertices of the network is given in the 1^{st} column. The number of centers and their corresponding optimal radii for the integer programming case are listed in columns 2 and 3, respectively. In the 4^{th} and 5^{th} columns, the number of centers and their corresponding optimal radii for the LP relaxation case of the proposed method are presented, respectively. In the 8^{th} and 9^{th} columns, the number of centers and their corresponding optimal radii for the LP relaxation case of the classical method are presented, respectively. The last columns under the proposed and classical methods present the percentage gap of the optimal radii between the corresponding LP relaxation and the IP formulation.

			Proposed Method			Classical Method		
		Integer Programming	LP Relaxation			LP Relaxation		
n	**p**	**Opt rad**	**p**	**Opt rad**	**Gap**	**p**	**Opt rad**	**Gap**
100	1	185	1	185	0.00%	1	185	0.00%
100	3	140	2.95	139	0.71%	2.95	139	0.71%
100	5	115.5	4.92	115.5	0.00%	4.92	115.5	0.00%
100	10	88	10	87	1.14%	10	87	1.14%
100	15	70.5	15	70.5	0.00%	15	70.5	0.00%
100	20	58.5	20	58.5	0.00%	20	58.5	0.00%
100	25	50.5	24.75	50.5	0.00%	24.75	50.5	0.00%
200	1	108	1	108	0.00%	1	108	0.00%
200	3	90.5	3	90.5	0.00%	3	90.5	0.00%
200	5	79.5	4.84	79.5	0.00%	4.84	79.5	0.00%
200	7	73	6.9	71	2.74%	6.9	71	2.74%
200	15	53	14.7	52	1.89%	14.7	52	1.89%
200	20	44.5	19.71	44	1.12%	19.7	44	1.12%
200	25	39.5	24.92	39	1.27%	24.88	39	1.27%
300	5	56	5	56	0.00%	5	56	0.00%
300	15	41.5	14.88	41.5	0.00%	14.88	41.5	0.00%
300	24	34.5	23.42	34.5	0.00%	23.41	34.5	0.00%
400	5	44.5	5	44.5	0.00%	5	44.5	0.00%
400	12	36	11.87	35.5	1.39%	11.87	35.5	1.39%
400	25	27.5	24.45	27.5	0.00%	24.45	27.5	0.00%
500	10	31.5	9.78	31.5	0.00%	9.57	31.5	0.00%
500	15	28	14.51	28	0.00%	14.83	27.5	1.79%
500	25	24	24.24	24	0.00%	24.1	24	0.00%
600	26	23.5	25.1	23.5	0.00%	25.1	23.5	0.00%
600	44	19.5	43	19.5	0.00%	43	19.5	0.00%
700	24	20	22.96	20	0.00%	22.96	20	0.00%
700	43	16.5	41.3	16.5	0.00%	41.3	16.5	0.00%
800	35	16	33.7	16	0.00%	33.7	15.5	3.13%
800	62	12.5	57.96	12	4.00%	57.96	12	4.00%
900	28	15.5	27.4	15.5	0.00%	27.4	15.5	0.00%
900	60	11.5	58.69	11.5	0.00%	58.69	11.5	0.00%
TN(81DV)	1	1065	1	1065	0.00%	1	1065	0.00%
TN(81DV)	3	542	3	542	0.00%	3	542	0.00%
TN(81DV)	5	417.5	5	417.5	0.00%	5	417.5	0.00%
TN(81DV)	10	267	10	267	0.00%	10	264.5	0.94%
TN(81DV)	15	191	14.72	191	0.00%	14.72	191	0.00%
Comp 100	10	88	10	83.5	5.11%	10	83.5	5.11%
Comp 100	15	70.5	15	70.5	0.00%	15	70.5	0.00%
Comp 100	25	50.5	24.75	50.5	0.00%	24.75	50.5	0.00%
Average					0.50%			0.65%
Maximum					5.11%			5.11%

Table 7 Computational Results for the LP relaxation and the IP formulation

The observations in the computational analysis and the comparison between the IP formulation and the LP relaxation are as follows:

- When we compare the proposed and the classical methods, there is no significant difference in terms of percentage gaps in optimal radii.

- For the proposed method, on the average there is 0.50% difference between the LP relaxation and the IP formulation in terms of the optimal radii.

- For the classical method, on the average there is 0.65% difference between the LP relaxation and the IP formulation in terms of the optimal radii.

- The maximum percentage gap in terms of optimal radii between the LP relaxation and the IP formulation for both of the methods is 5.11%.

Chapter 6

CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

In this thesis, we studied the absolute p-center problem over complete, incomplete and planar networks. We presented an algorithm and performed computational analysis for both absolute and vertex-restricted cases of the p-center problem and we became the first to solve large instances up to 900 vertices on the absolute p-center problem. Furthermore, in order to do a comparison between the classical and proposed methods, we wrote and implemented software code for both of the algorithms.

In the literature, the p-center problem is solved up to 1817 vertices [9]. But the algorithm proposed by [9] is based on the vertex-restricted p-center problem and it has 1817 points on its feasible set of potential centers. Our algorithm is based on solving the absolute p-center problem and we solved instances up to 900 vertices with more than 600000 points on its feasible set of potential centers.

Initially, we presented an exact algorithm on the absolute p-center problem in detail. The algorithm focuses on solving the p-center problem via a finite series of minimum set covering problems which differ in the construction from the traditionally used set cover problems in the literature.

We performed computational analysis on the proposed algorithm and compared it with the classical approach in the literature in terms of computational efficiency. In this comparison, we tested medium and large scale test problems obtained from the O.R library. In addition to these, we developed a Turkish road network according to the data obtained from satellite images. We verified the Turkish road network in terms of up-to-dateness and accuracy by investigating the most recent printed version of the road network of Turkey. We used the Turkish road network in order to do a computational analysis on a planar network based on realistic instances.

In order to do a comparison between the proposed and classical algorithms, we wrote and implemented C codes for both algorithms. The implementation consists of two interdependent C codes. The first code evaluates the intersection point-radius pairs whereas the second code solves a finite series of set covering problems.

We observed that the proposed algorithm outperforms the classical algorithm in terms of average CPU time requirement and average number of branch and bound nodes. In addition, the proposed algorithm performed better in terms of the average number of iterations.

In order to observe the differences between the vertex-restricted and absolute cases in terms of optimality, we performed a computational analysis for the vertex-restricted case as well. In order to do a comparison between the vertex-restricted and absolute cases of the p-center problem, we used the same instances that were used for the absolute p-center problem. We observed that on the average the absolute case of the problem is 5.20 percent better than

the vertex-restricted case and the maximum difference between the absolute radius and the vertex-restricted radius is measured to be 18.55 percent at optimality.

We did a computational comparison between a complete and an incomplete network. We observed that although there is no difference in terms of optimality, the complete network requires far more CPU time and number of variables than the incomplete network.

Next, we did a computational comparison between complete and planar networks. We tested the Turkish road network as a planar network and converted it to a complete network. According to the results, there is a huge difference in terms of memory requirement and number of variables between complete and planar networks.

In order to investigate the LP relaxation bounds for the IP formulations of the proposed and the classical methods, we replaced the binary constraints with non-negativity constraints and solved the LP relaxation form of the formulation. According to the results there is no significant difference between the LP relaxation and the IP formulation in terms of percentage gap of the optimal radii.

For future research directions, the preprocessing phase of the proposed algorithm could be modified by making some edge eliminations. These edge eliminations would be significant since it would directly decrease the number of variables, hence the performance of the algorithm would be even much better with respect to the classical approach. In addition to this, one can use column generation and lagrangian relaxation methods in order to reduce the memory requirement of the algorithm.

BIBLIOGRAPHY

1. Al-Khedhairi,A. and Salhi, S. (2005), "Enhancements to two exact algorithms for solving the vertex *p*-center problem". *Journal of Mathematical Modeling and Algorithms*, 4:129–147

2. Bozkaya, B. and Tansel, B. (1998), "A spanning tree approach to the absolute p-center problem", *Location Science*; 6:83–107.

3. Cristofides, N. and Viola, P. (1971), "The optimum location of multi-centers on a graph", *Operational Research Quarterly*; 22:145-154.

4. Daskin, M. (1995), *Network and discrete location: models, algorithms, and applications*, Wiley, New York.

5. Daskin, M. (2000), "A new approach to solving the vertex *p*-center problem to optimality: Algorithm and computational results". *Communications of the Operations Research Society of Japan*, 45:428–436.

6. Dearing, P. M. and Francis, R. L., (1974). "A minimax location problem on a network", *Transportation Science*; 8:333-343.

7. Dijkstra, E.W (1959), "A Note on Two Problems in Connexion with Graphs", *Numerische Mathematik 1*, 269-271.

8. Dvir, D. and Handler, G. Y. (2004), "The Absolute Center of a Network", Networks; 43(2):109-118.

9. Elloumi, S., Labbe, M., and Pochet, Y. (2004), "A new formulation and resolution method for the p-center problem". *INFORMS Journal on Computing*, 16:84–94.

10. Garfinkel, R., Neebe, A, and Rao, M. (1977), "The m-Center Problem. Minimax Facility Location", *Management Science*; 23:1133-1142.

11. Hakimi, S. L. (1964), "Optimal Locations of Switching Centers and the Absolute Centers and Medians of a Graph", *Operations Research*: 12: 450-459.

12. Hakimi, S. L. (1965), "Optimum distribution of switching centers in a communication network and some related graph theoretic problems", *Operations Research*; 13:462-475.

13. Hakimi, S. L., Schmeichel, E. F., and Pierce, J. G. (1978). "On p-centers in networks", *Transportation Science*; 12:1-15.

14. Handler, G.Y. "Minimax Location of a Facility in an Undirected Tree Graph" (1973), pp. 287-293.

15. Handler, G. Y. and Mirchandani, P. B. (1979), *Location on networks: theory and algorithms*. *Cambridge, MA: MIT Press*.

16. Ilhan, T. and Pınar, M. C. (2001), "An efficient exact algorithm for the vertex *p*-center problem". http://www.optimization-online.org/DB_HTML/2001/09/376.html

17. Kariv, O. and Hakimi, S. L. (1979), "An algorithmic approach to network location problems. Part I: The p-centers", *SIAM Journal on Applied Mathematics*; 37:513-538.

18. Minieka, E. (1970), "The m-center problem", *SIAM Review*; 12:138-139.

19. Minieka, E. (1977), "The Centers and Medians of a Graph", *Operations Research*; 25:641-650.

20. Minieka, E. (1981), "A polynomial time algorithm for finding the absolute center of a network", *Networks*; 11:351-355.

21. Mirchandani, P. B. and Francis, R. L. and (1990), "Discrete Location Theory". *New York, John Wiley & Sons.*

22. Özsoy, F. A. and Pınar, M. Ç. (2006), "An exact algorithm for the capacitated vertex *p*-center problem", *Computers and Operations Research*; 33: 1420-1436.

23. Tansel, B. C., Francis, R. L., and Lowe, T. J. (1983), "Location on Networks: A Survey. Part II: Exploiting Tree Network Structure", *Management Science*; 29 (4): 498-511.

24. Tansel B.C., Personal Communication, *Department of Industrial Engineering, Bilkent University, Bilkent 06800, Ankara, Turkey.*