EXACT DECOMPOSITION ALGORITHMS FOR NONLINEAR LOCATION AND HUB LOCATION PROBLEMS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

EMİNE GÜNDOĞDU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
INDUSTRIAL ENGINEERING

NOVEMBER 2018

Approval of the thesis:

EXACT DECOMPOSITION ALGORITHMS FOR NONLINEAR LOCATION AND HUB LOCATION PROBLEMS

submitted by **EMİNE GÜNDOĞDU** in partial fulfillment of the requirement for the degree of **Doctor of Philosophy in Industrial Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Yasemin Serin
Head of Department, **Industrial Engineering** _____

Prof. Dr. Sinan Gürel
Supervisor, **Industrial Engineering Dept., METU** _____

Examining Committee Members

Prof. Dr. Haldun Süral
Industrial Engineering Dept., METU _____

Prof. Dr. Sinan Gürel
Industrial Engineering Dept., METU _____

Assoc. Prof. Dr. Ayşegül Altın Kayhan
Industrial Engineering Dept., TOBB ETU _____

Assist. Prof. Dr. Sakine Batun
Industrial Engineering Dept., METU _____

Assist. Prof. Dr. Vedat Bayram
Industrial Engineering Dept., TEDU _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Emine GÜNDOĞDU

Signature :

ABSTRACT

EXACT DECOMPOSITION ALGORITHMS FOR NONLINEAR LOCATION AND HUB LOCATION PROBLEMS

Gündoğdu, Emine
Ph.D., Department of Industrial Engineering
Supervisor: Prof. Dr. Sinan Gürel

November 2018, 172 pages

Developing exact solution algorithms to solve difficult optimization problems is one of the most important subjects in the operations research literature. In this dissertation, we develop Benders decomposition based exact solution algorithms (BDTAs) for handling nonlinearity in three selected nonlinear integer location/hub location problems. The first and second problem include nonlinear capacity constraints, while in the last problem, both objective function and the capacity constraints are nonlinear. In our decomposition algorithms, we used problem specific, logic based feasibility and optimality cuts. In addition to BDTAs, we propose a MISOCP reformulation for solving the nonlinear integer model, which arises in wireless local area networks, to optimality by using commercial solvers. Our computational study demonstrates that the performance of MISOCP is better than that of Benders decomposition based algorithms. This reformulation is general for any convex objective function as long as the constraints have the same structure as those in the first problem that we studied in this dissertation. The second problem includes nonlinear constraints in which product of binary variables exist and we develop a branch-and-check algorithm with several enhancement steps. In the last problem, nonlinear terms in the objective function and constraints are handled in Benders decomposition scheme. Our computational study demonstrates that the performance of Benders decomposition type algorithm is better than that of commercial solvers for especially difficult instances.

Key words: Benders Decomposition, Mixed Integer Second Order Cone Programming, Branch-and-Check Algorithm, Nonlinear Location and Hub Location Problems

ÖZ

DOĞRUSAL OLMAYAN YER SEÇİMİ VE ANA DAĞITIM ÜSSÜ SEÇİMİ PROBLEMLERİ İÇİN KESİN AYRIŞTIRMA ALGORİTMALARI

Gündoğdu, Emine
Doktora, Endüstri Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Sinan Gürel

Kasım 2018, 172 sayfa

Zor optimizasyon problemlerini çözmek için kesin çözüm algoritmaları geliştirmek, yöneylem araştırması literatüründe önemli konulardan biridir. Bu tez kapsamında, seçilen doğrusal olmayan yer seçimi ve ana dağıtım üssü seçimi problemleri için Benders ayrıştırma algoritmasına dayanan kesin çözüm yöntemleri geliştirdik. Son ele alınan problemde, hem amaç fonksiyonu hem de kısıtlar doğrusal değil iken, ilk ve ikinci problem sadece doğrusal olmayan kısıtlar içermektedir. Önerdiğimiz ayrıştırma algoritmalarında, problemlere özgü olurluluk ve optimallik kesileri türettik. Benders ayrıştırma algoritmasına ek olarak, kablosuz yerel ağ tasarım problemi için karma tamsayılı ikinci dereceden konik bir formulasyon önerdik.

Deneysel çalışmalarımızda, konik formulasyonun performansının Benders ayrıştırma algoritması temelli algoritmaların performansından daha iyi olduğunu gösterdik. Bu önerilen formulasyon, ele alınan kısıtların yapısı aynı kaldığı sürece tüm konveks amaç fonksiyonlarının bulunduğu modeller için geçerlidir. İkinci problem, (0-1) karar değişkenlerinin birbirleriyle çarpılmasından oluşan doğrusal olmayan kapasite kısıtları içermektedir. Bu problem için, içinde çeşitli iyileştirme algoritmalarının kullanıldığı dal-kontrol algoritması önerdik.

Son problemde hem doğrusal olmayan amaç fonksiyonunu hem de doğrusal olmayan kısıtları Benders ayrıştırma algoritması içerisinde çözdük. İkinci ve son problem için önerdiğimiz Benders ayrıştırma algoritması temelli çözüm yaklaşımlarının performansının ticari çözücülerden daha iyi olduğunu gösterdik.

Anahtar kelimeler: Benders Ayrıştırma Algoritması, Karma Tamsayılı İkinci Dereceden Konik Programlama, Dal ve Kontrol Algoritması, Doğrusal Olmayan Yer Seçimi ve Ana Dağıtım Üssü Problemleri

To my mother Selvihan, my father Nevzat and my brother Çağlar…

ACKNOWLEDGEMENTS

I would like to express my special thanks to my supervisor Prof.Dr. Sinan Gürel for all his help, guidance and patience during five tough years. For especially more stressful periods in PhD education, he spent time to understand and help me.

I would like to sincerely thank to Assoc. Prof. Dr. Hakan Gültekin for his guidance, encouragement and being a really good friend to me for the last seven years. He always believes that I will be a good academician in the future. This is one of the most important motivations in my academic career.

In this dissertation, I inspired from the research ideas that I learned from Stochastic Programming course. I would like to thank to Assist. Prof.Dr. Sakine Batun for giving this course and valuable comments during my thesis as a committee member. I would also like to acknowledge Prof.Dr. Haldun Süral, Assoc. Prof. Dr. Ayşegül Altın Kayhan and Assist. Prof.Dr. Vedat Bayram for being a committee member of my dissertation and their valuable suggestions. I would like to thank to Prof. Dr. Murat Köksalan, Prof.Dr. Meral Azizoğlu, Assoc. Prof. Dr. Sedef Meral and Assist. Prof.Dr. Melih Çelik for their encouragements throughout my PhD education.

I'm indebted to my mother Selvihan and my brother Çağlar for their never-ending love, patience and encouragements throughout this study. Without my mother and brother, this dissertation could not be possible.

I continue to my academic career as a postdoctoral fellow at CIRRELT, which is one of the dreams of my life. I would like to sincerely thank to Assist. Prof.Dr. Sibel Alumur Alev and Assist. Prof.Dr. Duygu Taş Küten for their valuable suggestions and help during the application process for postdoctoral positions.

For the years between 2013-2017, this study was financially supported by TÜBİTAK under the program, 2211E. I would like to thank to TÜBİTAK for this financial support.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Location problems such as set covering, maximum coverage, the p-median, facility location and hub location problems have a significant place in operations research literature. These problems include strategic and/or tactical level location decisions made in different sectors. They have been often solved by using mathematical programming methods.

The network structure that occurs in location/ hub location problems can be seen in large-scale transportation, telecommunication, and cargo delivery systems. In these networks, some nodes could be specialized as hubs, and using them provides significant benefits to reduce the overall cost in the network. As mentioned in [19], the aims of using hubs are to reduce costs related to satisfying the demand, increase the efficiency of the service provided and cope with uncertainities.

Although most of the studies in the literature consider linear location/hub location problems, real life applications usually involve nonlinearities. Nonlinear functions may be necessary to model congestion or delay in the network, amount of backbone traffic, cost, technological restrictions, etc.

These nonlinear location problems can be formulated as mixed integer nonlinear programs (MINLP). Such kind of formulations are usually more difficult to solve than the linear counterparts. MINLPs combine discrete decisions and nonlinear functions. Generally, MINLPs cannot be solved as fast as MILPs, yet.

Some forms of convex functions and convex sets can be represented via second order conic programming constraints. This helps to reformulate some MINLPs as mixed

integer second order cone programs (MISOCP). In Chapter 3, we consider a nonlinear wireless network design problem and show that the problem can be reformulated as an MISOCP. So that, the problem can be solved by using commercial branch and bound solvers.

An alternative exact solution approach for discrete optimization is to use Benders decomposition. Benders decomposition was first proposed by [11] to solve mixed integer linear programs. In this decomposition, the overall model is decomposed into two parts: *master problem* and *subproblem*. Master problem includes a subset of decision variables (integer or binary variables) in the original problem. These variables are usually called complicating variables. The subproblem includes the remaining continuous variables and a given solution to the master problem. When the subproblem is a linear programming problem, the algorithm is called *classical Benders decompositon*.

In classical Benders decomposition method, the cuts which are added to the master problem for convergence, are generated by using the dual of the subproblem. These cuts could be either feasibility or optimality cuts depending on the status of the subproblem. The Benders decomposition can be implemented in either iterative way, where the master problem is resolved from scratch, or in a branch-and-cut framework. Solving the master problem only once in a branch-and-cut decreases computational time which is shown in several studies in the literature.

Although Benders decomposition is initially developed to solve MILPs, it has been extented to solve nonlinear programs (NLPs), integer programs (IPs), multistage problems and stochastic integer programs (SIPs). In these cases, subproblem can be a feasibility check, an integer programming problem or a mixed integer linear programming problem. We refer the reader to [65] for an extensive literature review on Benders decomposition algorithm, its relationship with other decomposition algorithms and improvement steps for the algorithm. In the literature, Benders decomposition algorithm can be improved by making several enhancements such as adding valid inequalities to the master problem, developing efficient solution strategies for solving master problem and subproblem, and using modified master problem in which some variables of the subproblem are included.

In the case where the subproblem includes binary or integer variables, which will be the case in those location problems considered in this dissertation, generating cuts by using duality may not be possible. If the master problem in a Benders Decomposition (BD) of a problem includes only binary variables, the optimality cuts in [54] can be used. These cuts can be employed for any problem as long as its subproblem can be solved for a given master problem solution. In the study [54], there are two main assumptions:

1. The master problem must only include binary variables.
2. The subproblem has complete recourse property. This means that the subproblem is always feasible for a given master problem's solution. Therefore, in the study [54], only generic optimality cuts are proposed.

When the master problem only includes binary variables and the subproblem is just a feasibility problem, only feasibility cuts are generated. To generate general feasibility cuts which are valid for the master problem including only binary variables, the feasibility cuts proposed by [20] can be used. These cuts are generally known as *combinatorial Benders* cuts.

The aim of adding feasibility cuts to the master problem is to cut off the current solution from the feasible region of the master problem. In this dissertation, we propose valid inequalities to improve the master problems for quadratic capacitated concentrator location problem (QCCLP) and quadratic capacitated hub location problem (QCHLP). We develop problem specific feasibility cuts for both QCCLP and QCHLP. For QCHLP, we also propose problem specific optimality cuts and compare its computational performance against general purpose optimality cuts.

By using *lazyconstraint callback function* in commercial solvers (e.g., IBM ILOG CPLEX), implementing Benders decomposition type algorithms in a branch-and-cut (B&C) has become widespread. In this dissertation, we develop exact Benders decomposition type algorithms for three nonlinear integer facility and hub location problems.

In Chapter 3, we study a nonlinear wireless local area network design problem with two variants (WLANDP1,WLANDP2). This problem was first studied by [42]. In

this problem, we are given a set of access points (facilities), a set of user terminals (customers) with demands and power levels that can be installed on given access points. The problem is to choose which access points to open at which power level and to assign each user terminal to an access point while capacity constraints are satisfied. This problem can be seen as a single assignment capacitated facility location problem in which the capacities of the facilities are also determined by the decision maker. Due to the technical constraints in wireless communication, capacity constraints for access points are nonlinear. The mathematical formulation of this problem includes binary variables and nonlinear constraints, so these properties make the problem difficult to solve.

Both variants (WLANDP1 and WLANDP2) of the problem have the same constraints. The only difference is that the objective function of WLANDP2 only includes the cost of installing power, while the objective function of WLANDP1 includes the cost of installing power on access points and the cost of assigning user terminals to the access points.

We show that the nonlinear capacity constraints can be expressed via second order conic inequalities. Therefore, both variants of the problem (WLANDP1 and WLANDP2) can be solved as MISOCP problems. MISOCP reformulations make it possible to solve WLANDP1 and WLANDP2 by using commercial solvers such as IBM ILOG CPLEX. To the best of our knowledge, this reformulation is the first closed form formulation which can be solved by existing mathematical programming solvers for WLANDP1 and WLANDP2. As long as the constraints remain the same, this MISOCP formulation is applicable for any problem with a convex objective function.

Furthermore, for WLANDP1, we propose an exact Benders type decomposition algorithm which is quite similar to the Integer L-shaped algorithm (ILS). ILS is a well known decomposition method for two stage stochastic integer programms. Our proposed algorithm can be seen as one version of ILS algorithm where the number of scenarios is one. The aim of developing a Benders type decomposition algorithm is to deal with the assignment cost term in the objective function and the nonlinear capacity constraints. In the master problem, we determine the selected power levels on

each access point. In the subproblem, we determine the assignments of user terminals to access points. Since the subproblem includes binary variables, we don't use the linear programming duality for the subproblem. We used feasibility cuts proposed by [42] and optimality cuts proposed by [54].

For WLANDP2, we propose a Benders decomposition type algorithm which is a special case of the algorithm proposed for WLANDP1. In this problem, the decomposition does not include optimality cuts since in this case the subproblem is a feasibility check problem. We used the same feasibility cuts as in WLANDP1 case.

For WLANDP1, we evaluate the performance of our proposed Benders type decomposition and MISOCP formulation. Computational study demonstrates that in the average solving MISOCP reformulation requires shorter CPU times than the Benders type decompositon algorithm. Also, for a given CPU time limit, MISOCP could solve more instances to optimum than Benders decomposition type algorithms. We also observed that MISOCP has a better performance than Benders decomposition for the difficult instances.

For WLANDP2, we compared the computational performance of MISOCP, Benders type decomposition algorithm and branch-and-Benders cut algorithm within the given time limit. MISOCP finds the optimal solutions for most of the instances. One of the important results conducted from computational study is that average number of feasibility cuts in the branch-and-Benders cut method is less than the number of cuts in the Benders type decomposition due to the strong feasibility cuts in branch-and-Benders cut method.

In Chapter 4, we consider a quadratic capacitated concentrator location problem (QCCLP) which arises in telecommunication network design. A concentrator or multiplixer is a device aggregating, compressing, forwarding and transfering data in the system. Concentrators can be seen as hubs or facilities in the facility/hub location problems in the literature. Similar to hubs, concentrators provide advantages in terms of cost in telecommunication systems. The QCCLP was first studied by [53]. In QCCLP, we are given a set of nodes and a traffic matrix between the nodes. The problem is to determine a subset of nodes as hubs and the assignments of non-hub nodes to hub nodes. To satisfy the demand of each origin-destination pair, at least

one of the hub locations must be visited. The objective function to minimize is the cost of installing hubs and traffic routing between non-hub nodes and hub nodes. The problem has nonlinear capacity constraints since the backbone traffic (traffic between concentrator pairs) is expressed by nonlinear terms.

When the capacity constraints are linear, QCCLP is reduced to a linear capacitated concentrator location problem (LCCLP). The LCCLP is a capacitated single assignment facility location problem and it is NP-hard. Therefore, QCCLP is NP-hard. For QCCLP, we propose a Benders decomposition type algorithm in which we develop problem specific feasibility cuts. We implement the decompositon in a branch-and-cut (B&C) framework. So, we call our method the branch-and-check algorithm (B&Ch).

As in classical Benders decomposition, we decompose the problem into two parts: master and subproblem. Master problem determines which nodes are selected as hubs, and the assignment of non-hub nodes to hub nodes. Master problem includes a linear relaxation of the nonlinear capacity constraints. Given a feasible solution for the master problem, in the subproblem phase we check if the nonlinear capacity constraints in QCCLP are feasible for this solution.

As the subproblem does not have any objective function, only the feasibility cuts are added to the master problem, when they are necessary. We propose two feasibility cuts called multiple feasibility cuts and extended feasibility cuts. To strengthen the master problem formulation, we also propose two alternative valid inequalities. Using alternative feasibility cuts and valid inequalities resulted in alternative branch and check algorithms.

For QCCLP, we compared the computational performance of our branch-and-check algorithms with MIQCP solver of IBM ILOG CPLEX. We observed that branch-and-check algorithms require shorter CPU times than CPLEX. Moreover, for given time limit it solves more instances than CPLEX. By using the branch-and-check algorithms, more instances are solved to optimum in shorter computational time. Branch-and-check can solve 100-node instances to optimum while MIQCP solver cannot find integer feasible solutions in given time limit. We also explored the effects of enhancement steps on the computational performance of branch-and-check algorithms. We

observed that adding valid inequalities to the master problem has significant effect on computational performance of the algorithm. For the instances where capacity constraints are tight, the effect of using valid inequalities in the master problem is higher.

In Chapter 5, we consider a quadratic capacitated hub location problem (QCHLP) which is a general variant of QCCLP. The problem has the same assumptions and constraints as QCCLP except that the QCHLP has backbone traffic cost in its objective function. So QCHLP has nonlinear terms in both its objective function and constraints.

For QCHLP, we propose a Benders decomposition type algorithm in which the problem is decomposed into master and subproblem. The master problem of this decomposition is the same as the master problem in the branch-and-check algorithm for QCCLP except that in the objective function, it includes auxiliary variables that represent the nonlinear backbone traffic cost terms. Different than the branch-and-check algorithm proposed for QCCLP, in the branch and check algorithm for QCHLP, optimality cuts must be included to the master problem when they are neccessary.

Implementation of a Benders decomposition type algorithm (BDTA) is the same as the branch-and-check algorithm except that when the subproblem is feasible we add optimality cuts to the MP. We propose a problem specific combinatorial optimality cut. As an alternative, we also used the optimality cut given by [54]. To the best of our knowledge, branch-and-check algorithm is the first Benders decompositon type solution approach developed for QCCLP. Similarly, integer Benders decomposition is the first Benders type decomposition for QCHLP.

We give the related literature review in Chapter 2. We give MISOCP reformulation, Benders decomposition type algorithm for WLANDP1 and WLANDP2 and their computational comparison results in Chapter 3. In Chapter 4, we develop a branch-and-check algorithm with several enhancement steps for QCCLP and give comparison results with MIQCP solver of CPLEX. We also discuss the effects of each enhancement step on the performance of the branch-and-check algorithm in this chapter. In Chapter 5, we develop an exact Benders decomposition type algorithm for QCHLP. We give concluding remarks and future research directions in Chapter 6.

CHAPTER 2

LITERATURE REVIEW

In this dissertation, we studied exact solution methods for solving WLANDP, QC-CLP and QCHLP which are all nonlinear integer optimization problems. Since the problems that we consider in this dissertation are nonlinear location and hub location problems, we first give the related literature review on location/hub location problems. Although the literature on hub location problem is too wide, we give the most relevant studies in which single allocation is considered and nonlinear terms occur in the objective functions or constraints of the models.

The algorithms that we develop in this dissertation are based on Benders decomposition in which the subproblem is not an LP. Therefore, we give some studies considering Integer L-shaped algorithm, logic-based Benders decomposition, and branch-and-check algorithm to give necessary background for the Benders decomposition algorithm with integer subproblems. In addition to decomposition algorithms, we develop a MISOCP reformulation for the problem discussed in Chapter 3. In this chapter, we also give the review of the recent studies related to the implementations of MISOCP on different problems.

2.1 Location/Hub Location Problems

As mentioned in [19], hub location problems have distinguishing features that can be summarized as follows:

- Demand values are defined between any origin-destination pair.

- Some nodes are selected as hubs in the network.
- Flows can pass through the hubs.
- Transferring flows by using hubs provides benefit in terms of cost.
- Most of the studies in the literature consider minimizing cost such as cost of opening hubs and routing cost in the network.

A general hub location problem can be defined as follows: we are given a complete graph that is $G = (V, E)$. In this representation, V denotes the set of all nodes, while edges are described by E. Distance between any nodes and the flow that must be sent from one node to another node are given parameters. Most of the studies in the literature consider Euclidean distance between the node pairs. The main decisions in this general hub location problem definition, are to determine the hubs and assignments of non-hub nodes to hub nodes to satisfy the demand of each node pair. When additional properties related to the problem are added to the definition, some different decisions may also be considered in the problem. These decisions could be determining the type or/and capacity of the hubs when multiple choices on capacity values exist.

Hub location problems can be classified by using several criteria such as assignment type (multiple & single) of non-hub nodes to hub nodes, capacitated/uncapacitated version of the hub, number of hubs that should be opened in the network. Different types of hub location problems have been extensively studied in the literature. We refer the readers to the surveys ([5], [15], [17], [38], [49], [63]) and a book chapter ([21]) for hub location problems and their solution methods.

Three problems that we studied in this dissertation are in the class of nonlinear integer optimization problems, we give related literature on nonlinear location and hub location problems in the Section 2.1.1.

2.1.1 Nonlinear Location/Hub Location Problems

The problems that we study in Chapter 3 and Chapter 4 include nonlinear terms in the objective function of their mathematical models. The nonlinearity consists of

multiplication of binary variables. The first study considering such kind of objective function for hub location problem is related to uncapacitated variant. The first model for the *uncapacitated p-median hub location problem* was proposed by [62]. The assumption in this problem is that completely interconnected hub links may exist. The developed model is very similar to the classical p-median problem with a different objective function including quadratic terms. Due to the complexity of handling quadratic terms in the objective function, most of the studies focus on heuristic algorithms in the literature. [62] gives two heuristics which are based on enumeration to solve the quadratic model. In one heuristic, each demand point is assigned to the nearest hub location while the other heuristic considers first or second nearest hub. The first heuristic finds the optimal solution for the special case where there is no nonlinear objective function term in the model. [48] considers heuristic algorithms: Tabu search and GRASP. Another tabu search algorithm is used in the study ([74]). Lower bounds for the objective function on quadratic hub location problems are considered in the study ([61]).

[67] consider the uncapacitated single allocation p-hub median problem in which no fixed cost exists in the objective function. They propose two procedures to find lower bounds for uncapacitated single allocation p-hub median problem. The first lower bounding method depends on Gilmore-Lawyer bound proposed for the quadratic assignment problem. The other method is based on developing a new MILP formulation and solve it by using Lagrangian relaxation.

In [18], an MILP formulation is proposed to solve uncapacitated single allocation p-hub median problem exactly. A new variable to linearize the quadratic terms in the objective function is introduced. The formulation is linear, however when the number of nodes in the network increases, it becomes difficult to solve this mathematical formulation. The reason is that the number of variables and constraints increases exponentially. In [37], a linear programming formulation in which fewer variables and constraints exist is proposed to solve the problem exactly. They also develop an efficient heuristic to solve the problems in reasonable amount of time. [75] consider the uncapacitated p-median hub location problem with two variants: multiple allocation and single allocation and propose a mixed integer linear programming formulation whose linear programming relaxations is tight. By using this approach, they are able

to solve some problem instances which are not solved to optimum before.

To find a lower bound for the objective function, the continuous relaxation of the model in [75] could be used. The bound obtained from this relaxation is better than the bound of the LP relaxation of the model proposed in [37]. The formulation developed by [37] includes three-index formulation. However, the mathematical formulation proposed by [75] require higher computational time.

Recently, [57] consider five different variants of single allocation hub location problems: *uncapacitated single allocation p-hub median problem*, *uncapacitated single allocation hub location problem*, *linear capacitated problem* and *single allocation hub location problem with congestion*. All models have nonlinear terms in their objective functions. To handle these nonlinearity issues, several linearization techniques are used. As the number of nodes increases, linearization methods result in huge number of variables in the models. The main assumption behind the method given in [57] is that the distance between any node pair is Euclidean. This property provides to construct a new linearization and row generation procedure. By taking the advantage of this method, they can solve large size instances such as 200 nodes to optimum .

All the problems that we consider in this dissertation are in the class of single allocation location and hub location problems, so that we give studies on single assignment hub location problems. As the decomposition algorithms include special variant of the single allocation hub location problem in the master problem, they have a significant place.

Although the uncapacitated variant of single allocation hub location problem is widely studied in the literature, studies on its capacitated counterpart appear relatively rare. [18] give the first linear integer programming formulation in which the variables are four-indexed for capacitated single allocation problem. To obtain a lower bound, in [8], a branch-and-bound method which is based on lagrangean relaxation is given. In [8], two heuristic algorithms are developed to obtain upper bounds. The similar problem is also considered in [9]. The difference is that the number of hubs that must be open is a given parameter in [9].

[53] consider several capacitated single allocation hub location problems in which the

hub capacity is used for both inflow & outflow and backbone traffic associated with the hub. Due to the amount of backbone traffic, the model proposed in [53] includes nonlinear constraints. They propose a nonlinear integer mathematical model in which a two indexed decision variable definition is used. Although the number of variables is fewer, the number of constraints is exponential. They study the polyhedral structure of the problems. They propose an exact branch-and-cut algorithm for the problems and seperate the exponential number of constraints in the branch-and-cut algorithm. They give the computational results for the instances up to 50 nodes. For QCCLP and QCHLP that are discussed in [53], we propose Benders decompositon type algorithms. To the best of our knowledge, our Benders decomposition algorithms are the first exact decomposition type algorithms that are implemented for both problems. We handle the nonlinear constraints (in QCCLP and QCHLP) and the nonlinear objective function (in QCHLP) within a Benders decomposition type approach. We demonstrate that our decomposition approach enables solving large size instances and also difficult instances to optimum.

[25] consider the capacitated hub location problem with single assignment. They show that using a Lagrangean relaxation technique, the problem can be decomposed into smaller subproblems which can be solved efficiently. In their lagrangean relaxation, the model does not have the integrality property, lower bound obtained from this relaxation is better than the LP-relaxation bound. In their algorithm, both lower and upper bounds are proposed.

[36] consider a capacitated single allocation hub location problem and give the formulations with fewer variables and constraints than the formulations in the literature. They develop two simple heuristics to obtain an upper bound. One of the heuristics is based on simulated annealing and the other one is based on random descent. They found the optimal solutions by solving a branch-and-bound algorithm and use upper bound in heuristics for a capacitated single allocation hub location problem. [24] develop a branch-and-price algorithm for CSAHLP and solve the instances optimally up to 200 nodes.

One variant to the QCHLP which is called *hub location problem with modular link capacity*, was studied by [79]. In this study, they consider a capacitated single alloca-

tion hub location problem that arises in telecommunication networks. Although the objective function of the proposed model is linear, the capacity constraints include nonlinear terms. As in QCHLP, the nonlinear terms are the multiplications of the binary variables. These terms are used for not allowing double counting in capacity constraints. They develop an exact and a heuristic method for *modular link hub location problem*. For the same problem, [26] propose a metaheuristic and compare the computational performance of the algorithm with the results given in [79].

For all problems in this dissertation, we handle nonlinear terms in a Benders decomposition framework. In addition to nonlinearity, the other difficulty in these problems is that single assignment is taken into consideration. Therefore, we give the related literature on exact solution approaches for single allocation hub location problem in Section 2.2.

2.2 Exact Solution Approaches for Single Allocation Hub Location Problems

Although many studies consider heuristic algorithms for single allocation hub location problems, exact algorithms are relatively rare in the literature.

In [30], a capacitated single allocation hub location problem is considered with two objective functions. They propose a bi-criteria model which handles the limitations that can occur in classical capacitated hub location problem. If some amount of flow exceeds the capacity of the hubs, excessive flow is rejected in the classical capacitated hub location problem. However, such a situation is not possible when an emergency service is considered. To model the problem as a bicriteria approach, the capacity constraints are considered as soft constraints. They develop bi-criteria approaches for single allocation hub location problem and find the nondominated solutions by using an interactive method. They give the solutions for 40-nodes instances.

In [29], hub capacities are determined by the decision maker in addition to the assumptions in classical CSAHLP. They propose several MILP formulations for the problem and compare the LP-relaxation bounds.

[51] also considers capacitated hub location problems and they used modified MILP

formulations. This reformulation include fewer number of variables and constraints rather than the formulations in literature. Two evolutionary algorithms are proposed for the problem and they could solve large size instances to optimum.

2.3 Benders Decomposition Approaches for Hub Location Problems

As solving uncapacitated variant of the hub location problems are easier than solving the capacitated versions, Benders decomposition algorithms are firstly implemented for uncapacitated hub location problems.

In [32], they implement different Benders decomposition variants for *UMAHLP*. The first exact method is a classical *Benders decomposition implementation* in which single cut is added to the master problem (*MP*) in each iteration of the algorithm. They also generate cuts for each origin-destination pair and add multiple cuts to the master problem. The last variant they use is that they stop the algorithm when an ϵ optimal solution is found. These three algorithms are implemented in an iterative way where master problem is resolved after adding new constraints to the *MP* through the iterations.

For a large scale uncapacitated multiple allocation hub location problem, a Benders decomposition algorithm is implemented in [22]. They implement Benders decomposition for the strong path based formulation. In order to generate strong optimality cuts, they use pareto optimal cuts. They generate multiple cuts for each candidate hub location. They could solve very large size of instances with 500 nodes optimally.

In the classical hub location problems, large discount factors for small amount of flow between interhub connections are used. In the study [34], a piecewise linear concave cost function between two hub locations is considered. They solve the problem by using Benders decomposition.

There are also studies in which Benders decompositon is applied for stochastic hub location problems. In the study [58], they propose MILP formulations for multiple allocation p-hub median problem. They consider hose and hybrid demand uncertainty. They develop exact solution methods based on Benders decomposition. In their com-

putational study, the performance of the decomposition algorithms is better than the commercial solvers. By using the decomposition algorithms, the instances that are not solved to optimum by using commercial solvers can be solved.

Capacitated multiple allocation hub location problem with hose demand uncertainty is introduced in [59]. They develop MIP formulation for the problem and two Benders decomposition algorithms are developed to solve the problem optimally. They implement their Benders decomposition algorithms in a single branch-and-bound tree of the master problem.

In [27], an integrated logistics network design problem, in which locations, capacities, mode selections for transportation decisions are made, is considered. They propose two approaches to solve the problem: a simplex-based branch and bound and a Benders decomposition approach. They conclude that Benders decomposition's performance is better than that of the other approach for difficult instances.

When the small number of hubs is considered in the network, congestion occurs in these hubs due to the amount of flow that is assigned to these hubs. [31] study a single allocation hub location problem under congestion. The aim is to minimize the total cost in the network. The objective function includes three terms: *installation of the hubs, congestion at hub nodes and routing cost*. They propose two MINLP formulations and a generalized Benders decomposition is used to solve these formulations exactly.

The Benders decompositon algorithms mentioned so far are classical Benders decomposition where the subproblem is an LP. To the best of our knowledge, there is only one study where an Integer L-shaped algorithm is implemented for a hub location problem. In [68], they study a single allocation hub location problem with hub breakdown. It is assumed that hubs cannot be available due to several reasons. For such a case, the amount of flow that is assigned to this hub can be assigned to backup hub. The mathematical model includes nonlinear objective function and they develop a branch-and-cut approach based on Benders decomposition, which is an Integer L-shaped algorithm.

2.4 Logic Based Benders Decomposition (*LBBD*)

In the BDT algorithm proposed in this dissertation, we develop problem specific feasibility and optimality cuts for the problems. The algorithms which use problem specific cuts are in the class of logic based Benders decomposition.

Logic based Benders decomposition was first developed by [45]. It is stated that the dual information (*inference dual*) must be obtained from any type of subproblems. If the subproblems are linear programs, the decomposition method is called classical Benders decomposition.

Logic based Benders decomposition can be seen as the general version of the classical Benders decomposition since the subproblem could be any optimization problem. One of the main differences between LBBD and classical Benders decomposition method is that no standard way exists to generate cuts in LBBD. Each feasibility and optimality cut generated is problem specific. *LBBD* is an iterative approach in which master problem is resolved after adding necessary cuts.

As stated in [10], the implementation of *LBBD* has three challenges defined as below.

- Adding valid inequalities which includes the information of subproblem
- Determining the time when subproblems are solved (e.g., solving the subproblems for each integer feasible solution or for an only optimal solution to the *MP*)
- Generating strong Benders cut (e.g., feasibility or optimality cuts)

LBBD is mainly used for scheduling problems in the literature. [28] consider the scheduling and routing of AGVs in the flexible manufacturing systems. In the problem, assignment and scheduling decisions are made simultaneously. They develop a hybrid decomposition algorithm to solve the problem, in which logic cuts are obtained from the subproblem.

In [46], a scheduling problem is considered. In this problem, each task must be assigned to a facility. Each task has a release and due date. They consider the problem

with different objective functions: *cost*, *makespan* and *total tardiness*. Assigning the tasks to the facilities are done using MILP and then they are scheduled by using CP. Therefore, they combine two approaches in logic based Benders decomposition.

In addition to scheduling problems, LBBD is also implemented for a location- allocation problem. In [39], they consider a location-allocation problem where the problem is to determine the locations of the facilities, assignments of the customers to the facilities without exceeding the capacities of facilities. In this problem, customers are assigned to the trucks at the facilities and each truck has a travel distance limit. Integer programming and constraint programming are used in a hybrid algorithm for this location-allocation problem.

There are also some studies in which nonlinearity is handled with logic based Benders decomposition. In [42], a wireless local area network design problem with two variants is proposed and a branch-and-Benders decomposition algorithm is implemented for both variants. They generate combinatorial feasibility cuts which use the special structure of the problem. Since generation of the cuts does not depend on the duality of the subproblem, valid cuts are generated logically. They give the computational comparison of Benders decomposition which is an iterative procedure and branch-and-Benders decomposition for the special case of two variants.

In Chapter 3, we consider this WLANDP with both variants and develop Benders decomposition type algorithms. In these decomposition algorithms, we used the problem specific feasibility cuts proposed in [42]. When optimality cuts are necessary for the convergence, we employed the logic behind Integer L-shaped algorithm whose definition will be given in Section 2.6.

Another example, where LBBD is used for dealing with nonlinearity of the constraints is given in [77]. They consider an integrated inventory-location problem with stochastic service constraints. A nonlinear mixed integer linear program is developed for this problem. Due to the structure of the decisions, the mathematical model only includes binary variables. Since the service requirement constraints are nonlinear, implementation of classical Benders decomposition is not posssible. They obtain a MP by removing the nonlinear service requirements. After solving the MP without considering the nonlinear constraints, they calculate the service level for each part

for the given master problem solution. If the subproblem is infeasible for the master problem's solution, they generate problem specific valid feasibility cuts and add them to the MP. They implement this LBBD algorithm in an iterative way.

2.5 Branch and Check Algorithms *(B&Check)*

Branch-and-check algorithm, which is a generalized variant of logic based Benders decomposition, was proposed by [76]. In LBBD, subproblem is solved after the optimal solution to master problem (MP) is found. However, the subproblem is solved for each feasible solution of the master problem in branch-and-check algorithm. The key difference between these two methods is that LBBD is implemented in an iterative way, while branch-and-check algorithm is implemented within a branch-and-cut scheme.

[10] gives the first comparison between LBBD and branch-and-check algorithm. In the paper, it is stated that the performance of the LBBD or branch-and-check algorithm depends on the difficulty of solving master and subproblem. When subproblems are more difficult to solve rather than MP, the performance of branch-and-check (B&check) algorithm could be weak.

Since MP is not resolved from scratch in branch-and-check algorithm, the performance of *branch-and-check* algorithm is generally better than an iterative approach.[76] give the name, branch-and-check to this algorithm and implement it on a planning and scheduling problem. A significant reduction on computational time is obtained when compared to the LBBD model proposed by [47].

In the study ([13]), subproblems are solved more often since the cuts are added to each node found in the branch-and-bound tree of master problem. They compare its approach with the one proposed by [47] and it's computational performance is better than the approach in [47].

The assignments of the jobs to the unrelated parallel machines under the cost minimization objective function is studied in the paper [70]. Each job has a release date and deadline that must be met in this problem setting. They implement seven different

branch-and-cut approaches which differ from each other in terms of formulation and master problem. In their implementation, the *MP* that they use in one branch-and-cut algorithm is tighter than the one given in [47]. In this paper, the efficiency of the computational performance of the algorithms that are not dominated comes from a tighter integer programming.

Branch-and-check algorithms differs from the logic based Benders decomposition in terms of implementation. In the decomposition algorithm that we propose for QCCLP, subproblem is just a feasibility check. Feasibility cuts are added to MP in the its branch and bound tree when they are neccesary. Therefore, we call our proposed Benders decomposition type algorithm branch-and-check algorithm in this dissertation.

2.6 Integer L-shaped Algorithm (ILS)

Integer L-shaped algorithm is a Benders decomposition algorithm which is widely used for two stage stochastic integer programs. This algorithm was proposed by [54]. Although the ILS algorithm was developed for two stage stochastic integer programs, this algorithm is applicable for any type of MIP formulations when the master problem has certain properties. MP must include only binary variables and subproblems could be every kind of optimization problems. ILS algorithm in [54] includes only optimality cuts due to the structure of the subproblems they assume. Generating feasibility cuts is not necessary for such kind of a subproblem since the subproblems are always feasible for a given master problem's solution. The optimality cuts which are generated in [54] are also known as *LL cuts*. However, as mentioned in ([71]), the same algorithm can be extented for the case in which both optimality and feasibility cuts must be generated. For the last problem in this dissertation, QCHLP, we used LL cuts to evaluate the performance of problem specific cuts. Although this kind of cut is proposed for the stochastic integer programs, we could use this general purpose optimality cut for WLANDP1 and QCHLP.

ILS algorithm has been used to solve many stochastic optimization problems optimally. In [55], they consider a capacitated vehicle routing problem where the demand

of some customers are stochastic. When the demand of a customer is not satisfied, this occurs a failure and the vehicle must go back to the depot. They implement an exact ILS method for the problem and show that some instances with 100 customers could be solved optimally. Some other implementations of Benders decomposition can be seen in several studies ([44],[60]).

2.7 Mixed Integer Second Order Cone Programming (*MISOCP*)

As given in [12], MISOCPs can be described as:

$$\begin{aligned} & \min c^T x \\ \text{(MISOCP)} \ \ & \text{s.t.} \ \ \|A_i x + b_i\| \leq a_{o_i}^t x + b_{o_i} \ \ \forall i \in 1, ..., m. \end{aligned} \tag{2.1}$$

where x represents the decision variable vector with n dimension and $\|A_i x + b_i\|$ denotes a Euclidean norm. x can include integer, binary or continuous variables.

The constraints that can be expressed in this form define second order cone or Lorentz cone. When the formulation (MISOCP) does not have any integer or binary variables, MISOCP is reduced to SOCP which are convex optimization problems. Several problem types can be formulated as SOCP (e.g., linear programs, quadratic programs with convexity, quadratically constrained convex programming).

A SOCP can be described as follows by dropping the integrality restrictions of the decision variables.

$$\begin{aligned} & \min c^T x \\ \text{(SOCP)} \ \ & \text{s.t.} \ \ Ax = b && (2.2) \\ & \qquad x \in K. && (2.3) \end{aligned}$$

In this formulation, K is a closed convex cone.

As SOCP is a special form of semidefinite programming (SDP), SOCP is the problem type which is between LP, quadratic programming and SDP. We refer the reader to [3] for the theory of SOCP, applications of it and algorithms that solve SOCP.

Interior point methods can be implementable for any type of constrainted optimization problems. Therefore, these methods are extended to solve SOCP problems in polynomial time. In each iteration of the algorithm, SOCP requires more computational time than linear programming and quadratic programs but less computational effort when compared to semidefinite programming. In other words, SOCP is the extensions of LP, while it is a specialized case of nonlinear programming. Several problems in the literature have been formulated as SOCP.

The interior point algorithms for NLPs could be used for solving MISOCP. But there is an assumption behind these algorithms is that the objective function and constraints should be twice differentiable.

Branch-and-bound algorithm to solve MISOCP is an algorithm where at each node an interior point algorithm, which is special for SOCP, is solved.

[3] give the functions that can be representable as SOCPs. We consider a WLANDP in which nonlinear capacity constraints exist in Chapter 3. The nonlinear function in the problem can be classified into one of the classes which is summarized in [3]. We have inequalities in which sum of linear fractions exist and these inequalities can be expressed as second-order cone inequalities. As our proposed reformulation includes binary variables, the resulting model is in the class of MISOCP.

Interior point methods for SOCP and its implementations for some applied problems and their reformulations are second order cone programming are given in [52].

In the literature, SOCPs and MISOCPs have a wide range of applications. Network design problems, portfolio optimization, scheduling, assembly line balancing, stochastic programs with chance constraints are some of these applications. Availability of SOCP solvers within off the shelf branch-and-bound software have supported the use of MISOCP formulations. [3] summarize several different conic representable sets and functions. [2] consider a machine-job assignment problem in which processing times are decision variables and manufacturing costs are nonlinear functions of processing times. They give a strengthened conic reformulation for the problem. [43] considers a network flow problem with congestion costs and shows that by using second order conic inequalities, the problem can be reformulated as a MISOCP. [12]

summarize several other MISOCP applications in the literature. To the best of our knowledge, our study whose results are given in Chapter 3 is the first reformulating the nonlinear capacity constraints in WLANDP1 and WLANDP2.

In [7], they study joint facility location and inventory management problems with different variants depending on the capacity of the facilities, retailer demand and lead time stochasticity. They reformulate all versions by using conic quadratic MILPs. To improve the performance of the formulations, they use valid inequalities. Using these valid inequalities strengthens the formulations.

In [50], a lot sizing problem where the processing time of the jobs are controlled by using extra cost. They add a novelty to the lot sizing problem as in classical lot sizing problem, jobs' times are constant parameters. The problem has nonlinear cost terms in its objective function. Instead of using a heuristic algorithm to solve the problem, they give a SOCP reformulation that can be solved by a commercial solver. They conclude that large size instances could be solved optimally by using this approach.

In [4], a stochastic disassembly line balancing problem in which task times are normally distributed is studied. The aim is to minimize the number of open stations. Each task must be assigned to exactly one work station. Due to the stochasticity of the tasks, the mathematical model includes chance constraints. In the study, these constraints are rewritten as second order cone inequalities. They propose seven formulations two of which are SOCPs. The remaining formulations are piecewise linear mixed integer programs. They compare alternative formulations in a computational study.

CHAPTER 3

A NONLINEAR WIRELESS LOCAL AREA NETWORK DESIGN PROBLEM *(WLANDP)*

In this chapter, we consider a nonlinear wireless local area network design problem with two variants. In this wireless network design problem, we are given a set of access points, power transmission levels applicable on the access points and user terminals with given demands. In the problem, two decisions should be made simultaneously. The first decision is which access points to open at which power (*capacity*) levels. The second decision is to assign each user terminal to an open access point. The solution should guarantee that the demand of each user terminal is satisfied and the capacity of the link defined between the access points and the user terminals is not exceeded. The objective function to minimize is the total power consumption at the opened access points.

The problem has similar properties with single assignment capacitated facility location problem. In WLANDP, while access points can be seen as the facilities, user terminals can be considered as the customers in a classical facility location problem. Selecting power levels in each access point can be seen as determining the capacity levels of the facilities in a facility location problem. Different than the similar problems in the literature in WLANDP, capacity constraint for each access point include nonlinear expressions.

We consider two variants which will be called WLANDP1 and WLANDP2. The difference between the two variants of the problem is that in the objective function, WLANDP1 includes an additional cost term for user terminal-access point assignments.

[40] handle nonlinear constraints for WLANDP2 by Benders decomposition, which is an iterative approach, while [42] implement a branch-and-Benders cut method for solving WLANDP1 and WLANDP2. We give three exact alternative methods for WLANDP2 and two alternative approaches for WLANDP1. For both WLANDP1 and WLANDP2, we compare the computational performance of the proposed solution methods.

Integer L-shaped algorithm (*ILS*) is one of the well-known exact algorithms to solve two-stage stochastic integer programs. ILS algorithm is a branch-and-cut approach based on Benders decomposition. Since the dual information cannot be extracted from the subproblem including binary or integer variables, the cuts in ILS algorithm does not depend on LP duality of the subproblem. As mentioned in [65], if Benders decomposition type algorithms are used to solve stochastic problems, they are called *L-shaped* algorithms.

ILS algorithm was introduced by [54]. In their paper, the main assumption is that master problem includes only binary variables and the subproblem has complete recourse property. In stochastic programming literature, if the subproblem is always feasible for a given master problem solution, it is said that the problem has complete recourse property. In the decomposition algorithms proposed to solve such kind of problems with complete recourse property, feasibility cuts are not generated.

[72] states that ILS algorithm is used for the problems in which the master problem only includes binary variables and the subproblem does not have complete recourse property. [54] develop general optimality cuts. In this study, we observe that the structure of the WLANDP *(e.g. integer MP and integer subproblem)* permits the use of an Integer L-shaped algorithm. As mentioned in [6], the Integer L-shaped method can be used for any mixed-integer programming problem if the objective function of the second stage (or subproblem) is calculated for a given binary solution for master problem. In the literature, Integer L-shaped algorithm was used as an exact solution method for several problems. Some examples can be seen in [56] and [60].

The computational study conducted for WLANDP1 demonstrates that MISOCP dominates the Benders decomposition type algorithm (e.g., ILS algorithm with only one scenario) in terms of CPU time and the number of instances solved optimally. For

WLANDP2, computational study shows that most of the instances are solved to optimum by using MISOCP. When the optimal solution is not found by any method, MISOCP gives a better integer solution. The aim of implementing a branch-and-Benders cut (*B&BC*) for WLANDP2 is to reduce the average number of feasibility cuts by using stronger cuts. We observe that the average number of feasibility cuts in *B&BC* is less than the number of feasibility cuts in Benders decomposition type algorithm (BDTA2).

This chapter is organized as follows. We give problem definition in Section 3.1. Exact solution techniques to solve WLANDP1 and WLANDP2 are given in Section 3.2 and Section 3.3. In Section 3.4, we give the computational study. We give concluding remarks in Section 3.5.

3.1 Problem Definition and Mathematical Formulation

The sets and parameters used in the mathematical formulation are as follows:

I : Set of user terminals.
J : Set of access points (APs).
K : Set of power levels in each access point.
p_0 : Fixed power consumption when an AP is powered on.
p_k : Power consumed when any AP at location $k \in K$ is used.
w_i : Demand of user terminal $i \in I$.
α_{ij} : The loss function parameter between i and j, where $0 \leq \alpha_{ij} \leq 1$.
μ_j : The power cost parameter related to each access point $j \in J$.

Below, we define two sets of decision variables in the problem.

$$x_{ij} = \begin{cases} 1, & \text{if the user terminal } i \text{ is assigned to access point } j \\ 0, & \text{otherwise} \end{cases}$$

$$y_{jk} = \begin{cases} 1, & \text{if the power level } k \text{ is selected at access point } j \\ 0, & \text{otherwise} \end{cases}$$

Here, we give the mathematical formulation of the problem proposed by [42].

$$\min \sum_{j\in J}\sum_{k\in K}(p_0+p_k)y_{jk} + \sum_{i\in I}\sum_{j\in J}\mu_j w_i x_{ij}$$

$$\text{(WLANDP1)} \quad \text{s.t.} \quad \sum_{j\in J} x_{ij} = 1 \;\; \forall i \in I \tag{3.1}$$

$$\sum_{k\in K} y_{jk} \leq 1 \;\; \forall j \in J \tag{3.2}$$

$$x_{ij} \leq \sum_{k\in K} y_{jk} \;\; \forall i \in I, \quad j \in J \tag{3.3}$$

$$\sum_{i\in I} \frac{w_i x_{ij}}{r_{ij}(\pi_j)} \leq 1 \;\; \forall j \in J. : r_{ij}(\pi(j)) > 0 \tag{3.4}$$

$$x_{ij} \in \{0,1\} \;\; \forall i \in I, \;\; j \in J \tag{3.5}$$

$$y_{jk} \in \{0,1\} \;\; \forall j \in J, \quad k \in K \tag{3.6}$$

When a power level k is selected on any access point j, the corresponding variable y_{jk} will be one. Then, we can calculate the radiated power level denoted by π_j for each access point j as below:

$$\pi_j = \sum_{k\in K} p_k y_{jk}.$$

If π_j is less than or equal to the given threshold value (γ_{ij}) for a user terminal-access point pair i-j, x_{ij} must be zero. If π_j is strictly greater than γ_{ij}, then x_{ij} could be either 1 or 0. In this formula, r_{max} is a maximum limit for the radiated power that can be sent to user terminals.

$$r_{ij}(\pi_j) = min\{r_{max},\ \pi_j \alpha_{ij}\} = min\{r_{max},\ \alpha_{ij}\sum_{k\in K} p_k y_{jk}\}.$$

To sum up, the $r_{ij}(\pi_j)$ function is described as follows:

$$r_{ij}(\pi_j) = \begin{cases} 0, & \text{if } \pi_j \leq \gamma_{ij} \\ min\{\alpha_{ij}\cdot\pi_j,\ r_{max}\}, & \text{otherwise} \end{cases}$$

In this formulation, constraint set (3.1) ensure that each user terminal must be assigned to exactly one access point. Constraint set (3.2) guarantee that at most one power level is selected at each access point j. Constraint set (3.3) satisfy logic relation between variables y_{jk} and x_{ij}. If no power level is selected at location $j \in J$, no x_{ij} variable can take value 1 for this specific location $j \in J$.

In Constraint (3.4), each term in the summation gives the ratio of capacity usage of a user terminal on each connection between user terminal i and access point j. The right hand side of this constraint is one in order to satisfy the capacity restrictions of the access point.

Lastly, constraints (3.5) and (3.6) are the binary restrictions.

To clarify the capacity constraints, we give the following example.

Example 3.1.1. *Consider a network with two open access points and four user terminals. Assume that user terminals 1 and 2 are assigned to an access point 1 and the remaining terminals 3 and 4 are assigned to the access point 2.*

The threshold values (γ_{ij}) on each connection between i and j are given as follows: $\gamma_{11} = \gamma_{12} = 40000$, $\gamma_{21} = \gamma_{22} = 45000$, $\gamma_{31} = \gamma_{32} = 60000$, $\gamma_{41} = \gamma_{42} = 55000$. *In this example, r_{max} value is set to 12000 and $\alpha_{ij} = 0.5$ for each i and j pair.*

Assume that $\pi_1 = 50000$ and $\pi_2 = 70000$. Consider a feasible assignment $\overline{x}$: $\overline{x}_{11} = 1$, $\overline{x}_{21} = 1$,$\overline{x}_{32} = 1$, $\overline{x}_{42} = 1$. *For this solution, we have the following inequality from Constraint set 3.4 for access point 1. The demand values of the user terminals are given by $w_1 = 100$ and $w_2 = 150$.*

$$\frac{w_1 \cdot \overline{x}_{11}}{r_{11}(\pi_1)} + \frac{w_2 \cdot \overline{x}_{21}}{r_{21}(\pi_1)} \leq 1.$$

In this inequality, $r_{11}(\pi_1) = \min\{\alpha_{ij}\pi_1, r_{max}\} = min\{50000(0.5), 12000\} = 12000$ and, $r_{21}(\pi_1) = 12000$.

By plugging these values in the inequality, we have:

$$\frac{100 \cdot 1}{12000} + \frac{150 \cdot 1}{12000} \leq 1.$$

The objective function includes two terms. The first term is total installation or power level selection costs at access points. The second term is the total power consumption incurred due to the amount of demand satisfied by an access point. If the parameter denoted by μ_j for each access point is equal for all access points or if there is no cost

term related to the the assignments of user terminals, the second part of the objective function can be removed, which gives WLANDP2.

3.2 MISOCP Reformulation of the Problem (*WLANDP1 and WLANDP2*)

In this section, we show that nonlinear capacity constraints (3.4) can be represented via second order conic inequalities. This helps to represent both problems (WLANDP1, WLANDP2) as MISOCP problems. We first introduce an auxiliary continuous variable (t_{ij}) to replace each term in constraint set (3.4).

After introducing t_{ij}, we can reformulate constraint (3.4) by the following two constraints (3.7) and (3.8).

$$\frac{w_i x_{ij}}{r_{ij}(\pi_j)} \leq t_{ij} \quad \forall i \in I, j \in J. \tag{3.7}$$

$$\sum_{i \in I} t_{ij} \leq 1 \;\; \forall j \in J. \tag{3.8}$$

By using the definition of the function $r_{ij}(\pi_j)$, we can replace constraint set (3.7) with the following two constraints.

$$\frac{w_i x_{ij}}{r_{max}} \leq t_{ij} \qquad \forall i \in I, j \in J \tag{3.9}$$

$$\frac{w_i x_{ij}}{\sum_{k \in K} y_{jk} p_k \alpha_{ij}} \leq t_{ij} \qquad \forall i \in I, j \in J \tag{3.10}$$

In other words, constraint set (3.4) is replaced with the constraints given in (3.8), (3.9) and (3.10). We define another parameter, $\beta_{ij} = \frac{w_i}{\alpha_{ij}}$, then constraint set (3.10) can be rewritten as follows:

$$\beta_{ij} x_{ij} \leq t_{ij} \pi_j, \quad \forall i \in I, j \in J. \tag{3.11}$$

When we take square of x_{ij}, the sign of the inequality does not change and we have the following set of constraints instead of constraint set (3.10).

$$\beta_{ij} x_{ij}^2 \leq t_{ij} \pi_j, \quad \forall i \in I, j \in J. \tag{3.12}$$

After reformulation of the constraints representing capacity of each access point, it is easy to see that constraint set (3.12) is in the class of hyperbolic constraints. By using this property, we can reformulate the constraints (3.12) as a second order conic inequality as below:

$$4 \cdot \beta_{ij} \cdot x_{ij}^2 + (t_{ij} - \pi_j)^2 \leq (t_{ij} + \pi_j)^2, \quad \forall i \in I, j \in J.$$

Given the conic reformulation of capacity constraints, an MISOCP reformulation of GWLANP2 is given below:

$$\min \sum_{j \in J} \sum_{k \in K} (p_0 + p_k) y_{jk} + \sum_{i \in I} \sum_{j \in J} \mu_j \, w_i x_{ij}$$

(MISOCP) s.t. $(3.1), (3.2), (3.3), (3.5), (3.6)$

$$\sum_{i \in I} t_{ij} \leq 1 \qquad \forall j \in J \tag{3.13}$$

$$\sum_{k \in K} p_k y_{jk} = \pi_j \qquad \forall j \in J \tag{3.14}$$

$$4 \cdot \beta_{ij} \cdot x_{ij}^2 + (t_{ij} - \pi_j)^2 \leq (t_{ij} + \pi_j)^2 \;\; \forall i \in I, \;\; j \in J \tag{3.15}$$

$$w_i \cdot x_{ij} \leq t_{ij} \cdot r_{max}, \quad \forall i \in I, j \in J \tag{3.16}$$

$$y_{jk} + x_{ij} \leq 1, \;\; \forall i \in I, \;\; j \in J, \;\; k \in K : p_k < \gamma_{ij} \tag{3.17}$$

$$t_{ij} \geq 0, \;\; \forall i \in I, \; j \in J \tag{3.18}$$

$$\pi_j \geq 0, \;\; \forall j \in J \tag{3.19}$$

MISOCP can be solved by using off-the-shelf optimization packages such as CPLEX. For WLANDP1 and WLANDP2, it is the first closed form formulation that can be solved by a solver. It is easy to implement compared to decomposition based approaches.

3.3 A Benders Decomposition Type Algorithm (*BDTA*)

In this section, we implement a Benders decomposition type algorithm which is mostly used to solve stochastic integer programming problems in the literature. The algorithm can be used to find optimal solutions for both problems (WLANDP1 and WLANDP2) with a slight difference. The relaxed version of the original nonlinear

model is the master problem in this decomposition. This relaxed problem is tightened by feasibility and optimality cuts when necessary. In WLANDP1, we need to consider both feasibility and optimality cuts, as the subproblem can be infeasible for a given master problem solution. We name the algorithms as BDTA1 for WLANDP1 and BDTA2 for WLANDP2. Note that, BDTA1 includes both feasibility and optimality cuts while BDTA2 contains only feasibility cuts. This algorithm is implemented in a single branch-and-bound tree of the MP.

3.3.1 A Benders Decomposition Type Algorithm for WLANDP1 (*BDTA1*)

- Master Problem in BDTA1 (M_y^{BDTA1})

  The master problem of this decomposition includes only the power level selection decisions (y_{jk}) and omits the capacity constraint set (3.4). Therefore, all the constraints in the master problem are linear. Also, the second term in the objective function of the original model is replaced with θ, which is an approximation term for the subproblem's objective function.

$$z(M_y^{BDTA1}) = \min \sum_{j\in J}\sum_{k\in K}(p_0 + p_k)\cdot y_{jk} + \theta$$

$$\text{s.t.} \sum_{k\in K} y_{jk} \leq 1, \qquad \forall j \in J. \tag{3.20}$$

$$\sum_{j\in J}\sum_{k\in K} y_{jk}\cdot p_k \cdot \alpha_{ij} \geq w_i \ , \ \forall i \in I. \tag{3.21}$$

$$y_{jk} \in \{0,1\}, \qquad \forall j \in J,\ k \in K. \tag{3.22}$$

$$\theta \geq 0. \tag{3.23}$$

  Constraints (3.21) guarantee that at least $\sum_{k\in K} y_{jk}p_k\alpha_{ij}$ amount of power should be radiated to satisfy the demand of the user terminal i. By adding these constraints, we avert the situation in which all y_{jk} values are 0. The valid inequality (3.21) given in [42] is added to the (M_y^{BDTA1}) to improve the performance of the algorithm.

  M_y^{BDTA1} is solved in a branch-and-cut framework. If an integer feasible solution is found in B&B node, then the subproblem is solved. If required, feasi-

bility and optimality cuts are added to the current B&B node. In other words, we interrupt the branch and bound algorithm when we find an integer feasible solution for M_y^{BDTA1} and solve the subproblem at that node.

We refer the reader to [69] for the rationale behind the implementation of Benders decomposition algorithm in a branch-and-cut framework by using callback functions available in commercial solvers.

- Subproblem in BDTA1 ($S_1(\overline{y})$)

  Suppose that an integer feasible solution $\overline{y} = \{\overline{y}_{jk} | \ \forall j, k\}$ is found for M_y^{BDTA1}. Then the following subproblem is solved:

$$z(S_1(\overline{y})) = \min \sum_{i \in I} \sum_{j \in J} \mu_j \cdot w_i \cdot x_{ij}$$

$$S_1(\overline{y}) \ \text{s.t.} \ \sum_{j \in J} x_{ij} = 1, \ \ \forall i \in I. \tag{3.24}$$

$$x_{ij} \leq \sum_{k \in K} \overline{y}_{jk} \ \ \forall i \in I, \ j \in J. \tag{3.25}$$

$$\sum_{i \in I} \frac{w_i \cdot x_{ij}}{min\{\overline{\pi}_j \cdot \alpha_{ij}, r_{max}\}} \leq 1 \ \ \forall j \in J : \overline{\pi}_j > 0. \tag{3.26}$$

$$x_{ij} = 0, \ \ \forall i \in I, j \in J, k \in K : p_k < \gamma_{ij}. \tag{3.27}$$

$$x_{ij} = 0, \ \ \forall i \in I, j \in J, k \in K : \overline{\pi}_j = 0. \tag{3.28}$$

$$x_{ij} \in \{0, 1\}, \ \ \forall i \in I, \ \ j \in J. \tag{3.29}$$

  $S_1(\overline{y})$ determines the assignments (x_{ij}) of user terminals to access points in solution $\overline{y}$.

  Constraint (3.25) satisfy that a user terminal can be assigned to an access point when the access point is open or a power level is selected at this access point. Constraint (3.26) is linear, since the value of $\overline{\pi_j} \cdot \alpha_{ij}$ is determined in M_y^{ILS1} for all i and j. If the selected power level is less than the threshold value between user terminal i and j, the corresponding assignment variables (x_{ij}) are set zero by Constraint set (3.27). Moreover, constraint (3.28) ensure that if no power level is selected for an access point $j \in J$ (e.g., the total power level is zero), then no user terminal is assigned to this AP.

- Feasibility and Optimality Cuts

When $S(\overline{y})$ is solved at a B&B node, two cases can occur:

CASE 1: Subproblem is infeasible: Assume that we find an integer feasible solution for M_y^{BDTA1}. In the subproblem, infeasibility can occur due to the capacity and single source assignment constraints. $\overline{y}$ must be cut off from the feasible region of M_y^{BDTA1} which can be done by adding the following cuts proposed by [42] to the M_y^{BDTA1}:

$$\sum_{j \in J: \overline{Y}_j = 0} \sum_{k \in K} y_{jk} + \sum_{j \in J: \overline{Y}_j = 1} \sum_{k \in K: p_k > \overline{\pi}_j} y_{jk} \geq 1. \tag{3.30}$$

where $\overline{Y}_j$ is defined as follows: $\overline{Y}_j = \sum_{k \in K} \overline{y}_{jk}, \forall j \in J$.

Inequality (3.30) makes sure that power level on at least one AP is changed.

CASE 2: Subproblem is solved to optimum: Consider that in the subproblem, we find a feasible assignment for $\overline{y}$. In order to close the gap between the θ value and the objective function value of the subproblem $z(S(\overline{y}))$, we need to add optimality cut to M_y^{BDTA1}.

These cuts use $\overline{y}$ and the objective function value of the subproblem. The general purpose optimality cuts proposed by [54] are given below:

Let's define two sets, S and $\overline{S}$ as follows:

$S = \{(j,k) : \overline{y}_{jk} = 1\}$ and $\overline{S} = \{(j,k) : \overline{y}_{jk} = 0\}$

$$\theta \geq (\theta(\overline{y}) - L)\{\sum_{j \in J} \sum_{k \in K: \overline{y}_{jk} = 1} y_{jk} - \sum_{j \in J} \sum_{k \in K: \overline{y}_{jk} = 0} y_{jk} - |S| + 1\} + L \tag{3.31}$$

where L is a lower bound for θ.

An L value can be calculated by considering a solution in which each user terminal is assigned to the cheapest (minimum μ_j) access point. Then, we obtain $L = min_{j \in J}\{\mu_j\} \sum_{i \in I} w_i$. In the optimality cut 3.31, $\theta(\overline{y})$ is $z(S(\overline{y}))$, which is the objective function value of $S(\overline{y})$. Moreover, in order to clarify, the flowchart of the overall BDTA1 algorithm is given in Figure 3.1. Note that, we give this figure for the problems where we can find an integer feasible solution from MP.

3.3.2 Benders Decomposition Type Algorithm for WLANDP2 (*BDTA2*)

We implement a Benders Decomposition Type Algorithm (BDTA2) for WLANDP2 and this algorithm is a simplified version of *BDTA1* where there is no optimality cut generated. The reason is that we do not have any objective function in the subproblem of BDTA2. For a given master problem solution, if the subproblem is infeasible, we generate the cuts proposed by [42] and add these cuts to the master problem.

3.3.3 Branch and Benders Cut Algorithm for WLANDP2 *(BBC)*

So far, we have discussed two solution methods, namely BDTA2 and MISOCP for WLANDP2. In this section, we present an alternative Benders decomposition type algorithm for the problem.

The master problem of BBC is given below:

$$z(M_{x,y}^{BBC}) = \min \sum_{j\in J}\sum_{k\in K}(p_0 + p_k)\cdot y_{jk}$$

$$\text{s.t.} \sum_{k\in K} y_{jk} \leq 1, \ \ \forall j \in J. \tag{3.32}$$

$$\sum_{j\in J} x_{ij} = 1, \ \ \forall i \in I. \tag{3.33}$$

$$x_{ij} \leq \sum_{k\in K} y_{jk}, \ \ \forall i \in I, \ \ j \in J. \tag{3.34}$$

$$\sum_{i\in I} \frac{w_i \cdot x_{ij}}{\alpha_{ij}} \leq \sum_{k\in K} p_k \cdot y_{jk}, \ \ \forall j \in J. \tag{3.35}$$

$$y_{jk} \in \{0,1\}, \ \ \forall j \in J, \ k \in K. \tag{3.36}$$

$$x_{ij} \in \{0,1\}, \ \ \forall i \in I, \ \ j \in J. \tag{3.37}$$

The master problem in branch-and-Benders cut decomposition (BBC) is a relaxation of the original model for WLANDP2. Constraint set (3.35) is obtained by relaxing the capacity constraints (3.4). In order to replace $r_{ij}(\pi_j)$ function in Constraint (3.4), we can use the upper bound for this function given in [42].

$$r_{ij}^{u}(\pi_j) = \alpha_{ij} \sum_{k\in K} p_k y_{jk}.$$

When this upper bound is used, a valid inequality that is obtained from the capacity constraints (3.4) is given as below:

$$\sum_{i \in I} \frac{w_i x_{ij}}{\alpha_{ij} \pi_j} \leq 1 \;\; \forall j \in J.$$

When $\pi_j = \sum_{k \in K} p_k \cdot y_{jk}$ is plugged in the inequality, we have a valid inequality as:

$$\sum_{i \in I} \frac{w_i x_{ij}}{\alpha_{ij}} \leq \sum_{k \in K} p_k y_{jk}, \;\; \forall j \in J.$$

Therefore, the relaxed version of the Constraint set (3.4) is added to the $M_{x,y}^{BBC}$. The aim of adding Constraint (3.35) to the master problem is to obtain better solutions from $M_{x,y}^{BBC}$. The subproblem of this decomposition is given as follows:

$$z(S_2(\overline{y})) = min \;\; 0 \tag{3.38}$$

$$S_1(\overline{y}) \;\; \text{s.t.} \;\; \sum_{j \in J} x_{ij} = 1, \;\; \forall i \in I. \tag{3.39}$$

$$x_{ij} \leq \sum_{k \in K} \overline{y}_{jk} \;\; \forall i \in I, \; j \in J. \tag{3.40}$$

$$\sum_{i \in I} \frac{w_i \cdot x_{ij}}{min\{\overline{\pi}_j \cdot \alpha_{ij}, r_{max}\}} \leq 1 \;\; \forall j \in J : \overline{\pi}_j > 0. \tag{3.41}$$

$$x_{ij} = 0, \;\; \forall i \in I, j \in J, k \in K : p_k < \gamma_{ij}. \tag{3.42}$$

$$x_{ij} = 0, \;\; \forall i \in I, j \in J, k \in K : \overline{\pi}_j = 0. \tag{3.43}$$

$$x_{ij} \in \{0, 1\}, \;\; \forall i \in I, \;\; j \in J. \tag{3.44}$$

Similar to BDTA2 algorithm, the master problem $M_{x,y}^{BBC}$ is solved by using the B&B algorithm. Once an integer feasible solution is obtained at B&B node, a subproblem $S_2(\overline{y})$ is solved. $S_2(\overline{y})$ is the same problem as $S_1(\overline{y})$ except that $S_2(\overline{y})$ has no objective function to minimize and it is just a problem of checking the feasibility of solving vector $\overline{y}$.

Depending on the solution status of the subproblem, we have two cases:

- *Case 1:* $S_1(\overline{y})$ detects a feasible solution, then an integer feasible solution is found for WLANDP2.
- *Case 2:* $S_1(\overline{y})$ is infeasible, we solve the subproblem, $S_2(\tilde{y})$. $\tilde{y}$ is a solution vector obtained from $\overline{y}$ as given below:

$$\tilde{y}_{jk} = \begin{cases} 1, & \text{if } k = k_{max} \text{ and } \sum_{l \in K} \overline{y}_{jl} = 1 \\ 0, & \text{otherwise} \end{cases}$$

After solving $S_2(\tilde{y})$, two cases can occur:

- *Case 2.1:* If the $S_2(\tilde{y})$ is feasible, it means we can find a feasible assignment vector, x_{ij} for $\tilde{y}$ which is obtained by increasing the power levels of opened access points in $\overline{y}$. Then, the following feasibility cuts can be added to $M_{x,y}^{BBC}$.

$$\sum_{j \in J: \overline{Y}_j = 1} \sum_{k \in K: p_k > \overline{\pi}_j} y_{jk} \geq 1. \tag{3.45}$$

- *Case 2.2:* If the $S(\tilde{y})$ is infeasible, then at least one of the access points which is not opened in $\overline{y}$ must be opened.

  To satisfy this, the following cut can be added to $M_{x,y}^{BBC}$:

$$\sum_{j \in J: \overline{Y}_j = 0} \sum_{k \in K} y_{jk} \geq 1. \tag{3.46}$$

  Both BBC and BDTA2 algorithms are Benders type decomposition algorithms. They are both implemented in a branch-and-cut framework. However, they have differences in their master problems and feasibility cut generation procedure. BDTA2 algorithm only includes y_{jk} variables in the master problem, while BBC includes both decision variables. In terms of the master problems in the decomposition algorithm BBC and BDTA2, the master problem in BBC gives a better solution with respect to the solution in BDTA2. The reason is that $M_{x,y}^{BBC}$ includes decision variables of the subproblem. Both BBC and BDTA2 include the same feasibility cuts proposed in [42].

  This algorithm is inspired by the one in [42]. However, they relax the integrality restriction on assignment variables (x_{ij}). It is expected that total number of feasibility cuts in this approach is less than those in BDTA2, since the master problem includes the decision variables of the subproblem. In other words, master problem in BBC includes more information than MP of BDTA2. The flow chart of BBC is given in Figure 3.2 in Page 44.

Previous work in the literature: The problem in this study was first presented by [42]. There are also two previous studies [40] and [41] of that paper. In [40], they only consider a specific version of the problem called as *without user terminal cost* (we call this problem as WLANDP2 in this paper). They implemented a Benders decomposition where feasibility cuts are combinatorial cuts. This implementation does not have optimality cuts, since the subproblem is just a feasibility problem. They take an *iterative approach* to solve the problem, where the master problem (MP) is resolved after adding feasibility cuts. The strategy for partitioning the variables is the same as the classical Benders approach. In other words, the variables of the subproblem are not included in the subproblem.

For more general version of the problem *with user terminal costs*, *WLANDP1*, they proposed a Branch-and-Benders cut approach. They decompose the model in a different way from the classical Benders decomposition. MP includes both master and subproblem's variables. They also relax the integrality of the subproblem's variables (*allocating of user terminals to access points*) in the master problem. Since the capacity relation of access points is nonlinear, they relax these constraints by using an upper bound for power consumption. The other main difference from classical Benders decomposition techniques is that they solve an auxiliary subproblem to determine the feasibility cuts that must be added to the master problem. This step is for strengthening the feasibility cuts and reduce the total number of cuts until convergence. In this implementation, the optimality cuts that are added to the current node, when the subproblem is feasible, do not depend on the objective function value of the subproblem.

The differences of the algorithms proposed by [42] and our methods are given in Tables 3.1 and 3.2.

3.4 Computational Study

In this chapter, we propose two exact solution methods: MISOCP and BDTA1 for WLANDP1 and three exact solution methods: MISOCP, BDTA2 and BBC for WLANDP2.

Table 3.1: Comparison of solution methods for WLANDP1 in the literature and our work

	WLANDP1	
	Decomposition	**Closed form formulation**
Gendron et. al[42]	**BBC** (MP- x_{ij} and y_{jk}) Two IP subproblems MP:($0 \leq x_{ij} \leq 1$)	—
This study	**BDTA1** MP (y_{jk}) ; SP (x_{ij}) MP and SP's are IP Feasibility cuts (3.30) Optimality cuts (3.31)	**MISOCP**

Table 3.2: Comparison of solution methods for WLANDP2 in the literature and our work

	WLANDP2	
	Decomposition	**Closed form formulation**
Gendron et.al [42]	**BBC** in Table 3.1 Iterative Approach: **[40]** Multiple master problem solutions	—
This study	**BDTA2** with feasibility cuts (3.30) ***BBC (Master and subproblems:IP)***	**MISOCP**

We give computational comparison of proposed methods for each problem.

In Table 3.3, we give the experimental factors and their levels used in our computational study for WLANDP1 and WLANDP2.

We have generated problem instances with different levels of number of user terminals ($|I|$), number of candidate access points ($|J|$), maximum power limit on each node pair (r_{max}) as given in Table 3.3. Demand values of user terminals (w_i) are randomly generated from the interval given in Table 3.3 as low and high. γ and μ are randomly generated from the intervals given in Table 3.3, so there are 24 treatment combinations. For each combination, five random instances were generated.

In our experiments, we assumed that four power levels are available on each access point. The parameter values for p_0 and p_k, $\forall k = 1, .., 4$ are randomly generated from the intervals given in Table 3.3.

We conduct our experimental runs on a 64-bit machine @ 1.60 GHz and 4 GB of Ram. All methods are coded in IBM CPLEX 12.6.2 using C++ Concert technology.

Table 3.3: Factors and their levels in the computational study

Factors	Levels
$\|I\|$	150 (L)
	200 (H)
$\|J\|$	10 (L)
	15 (H)
w	U[100,150] kbps (L)
	U[200,250] kbps (H)
γ	U[40,000,60,000]
μ	U[30,50]
r_{max}	8,000 (L)
	12,000 (M)
	20,000 (H)
p_0	U[100,000,600,000]
p_1	U[10,000,30,000]
p_2	U[30,000,50,000]
p_3	U[50,000, 70,000]
p_4	U[70,000,90,000]

To implement BDTAs, we need to construct a single tree for the master problem and solve subproblem at each node of that tree. So, we use *Ilolazyconstraint callback* function that is available in IBM ILOG *CPLEX*.

3.4.1 Comparison of MISOCP and BDTA1 for WLANDP1

We first test the computational performance of MISOCP and BDTA1 for WLANDP1. One of the performance measures is the solution time in CPU seconds. We set a time limit of 3600 CPU seconds for each method. We give the computational comparison results of MISOCP and BDTA1 in Table 3.4.

In this table, the 'CPU' columns give the average CPU times obtained from five random instances for each setting. The average CPU time also includes the CPU time of the instance, which is not solved to optimum in given time limit.

In Table 3.4, 'Nodes' column under the tab of MISOCP represents the average number of nodes opened in the branch and bound tree. Similarly, the 'Nodes' column in BDTA1 shows the average number of opened nodes in branch and bound tree of master problem (M_y^{BDTA1}).

The 'gap_1' column under MISOCP method is calculated as

$$gap_1 = 100 \cdot (Z_{obj}^{MISOCP} - Z_{best}^{MISOCP})/Z_{best}^{MISOCP},$$

where Z_{obj}^{MISOCP} is the best lower bound achieved at the end of time limit and Z_{best}^{MISOCP} is the objective function value of the best integer solution found in the branch and bound tree.

The 'gap_2' column under BDTA1 is defined as

$$gap_2 = 100 \cdot (Z_{obj}^{BDTA1} - Z_{obj}^{MISOCP})/Z_{obj}^{MISOCP},$$

where Z_{obj}^{BDTA1} is the objective function value of the master problem at termination due to time limit.

In Table 3.4,we also report the number of APs opened in the optimal solutions. We obtain the number of APs opened in the solution from one of the algorithms that finds the optimal solution. If the solution is not optimal at the end of time limit, we report the number of current open APs in the best solution. In Table 3.4,*'Opt'* column under each method represents the number of instances solved to optimum within the time limit. The last two columns of Table 3.4 give the total number of cuts and the percentage of feasibility cuts in total number of cuts. The last column (*Feas (%)*) is calculated as (the number of feasibility cuts/ total number of feasibility and optimality cuts)$\times$100.

When we compare the performance of BDTA1 and MISOCP in terms of CPU times, we can see that MISOCP outperforms BDTA1. Even though, there are some instances in which BDTA1 solves faster, MISOCP is always better on the average CPU time for five instances.

Over 120 instances, MISOCP solves 97 instances to optimum while BDTA1 solves 55 instances to optimum. We can say that MISOCP outweighs BDTA1 in terms of number of optimally solved instances as well. Detailed results given in Tables 2.1, 2.2 and 2.3 of Appendix B show that there is no setting where BDTA1 can solve the problems that are not solved to optimum by MISOCP.

Except the setting (*treatment combination*) 18, all gap_2 values are non-negative. This

means that in almost all instances where both methods failed to find an optimal solution, MISOCP found better integer feasible solutions.

Since $r_{ij}(\pi_j)$ function value decreases with respect to r_{max}, the capacity usage of a user terminal on the open APs increases. As a result of this, smaller number of user terminals can be assigned to the same AP. In the WLANDP, all demand of user terminals must be met. To satisfy this requirement, the number of APs opened is high when r_{max} value becomes smaller. This implies that adding more feasibility cuts in BDTA1 is necessary. As the number of cuts increases, the required computation time of the BDTA1 increases. The instances where r_{max} value is small are difficult to solve optimally by BDTA. Moreover, the number of instances solved to optimum is less than the one in MISOCP when r_{max} value is small.

The value of r_{max} also has an impact on the MISOCP reformulation. Since the instances become more difficult when r_{max} value decreases, the number of nodes opened in the $B\&B$ tree of MISOCP increases. As can be seen in Table 3.4, the highest value on the number of nodes occur in the setting with $r_{max} = 8000$. Maximum gap_1 value achieved by MISOCP is (8.7%).

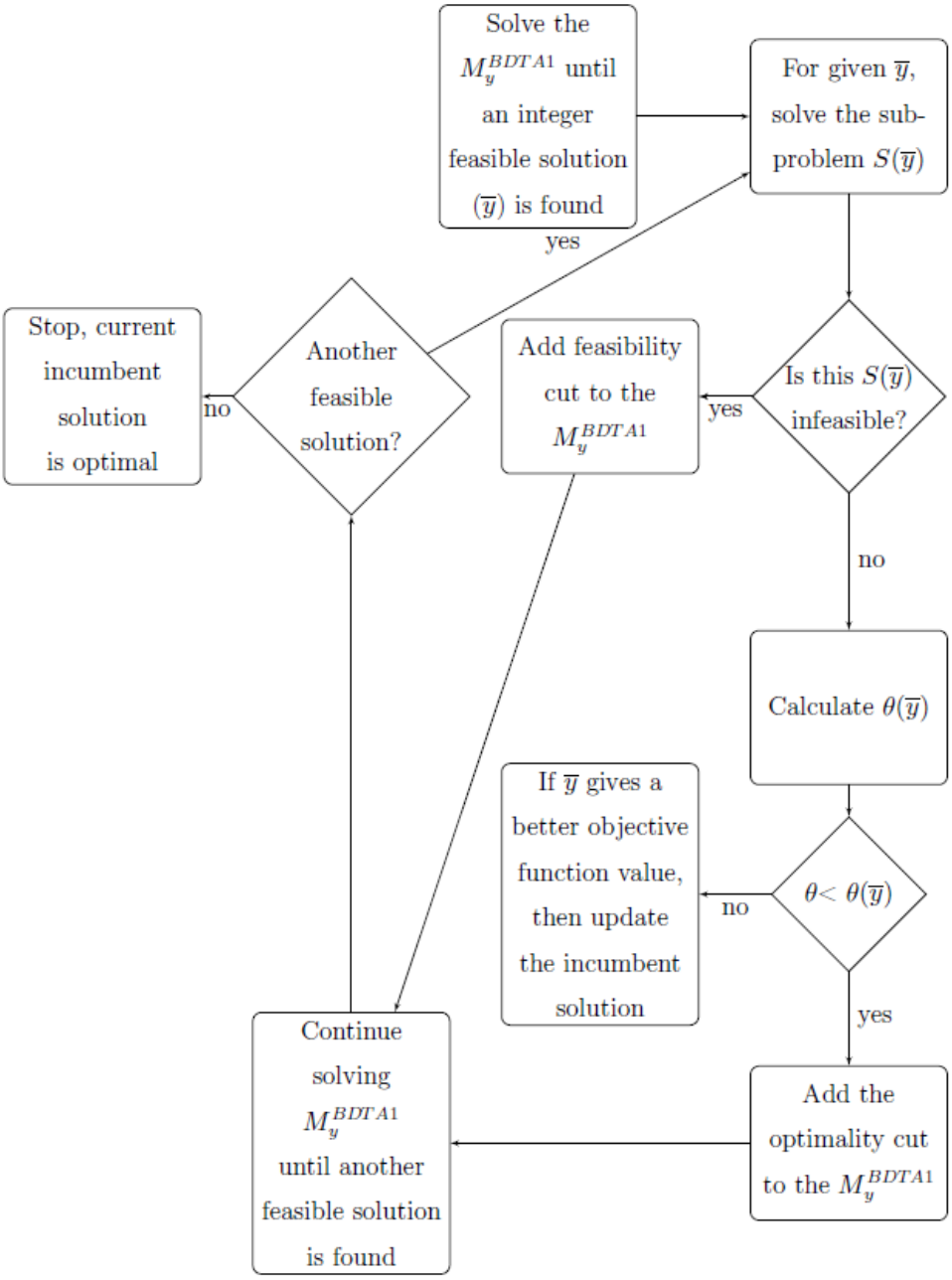


Figure 3.1: Flowchart of BDTA1 for WLANDP1

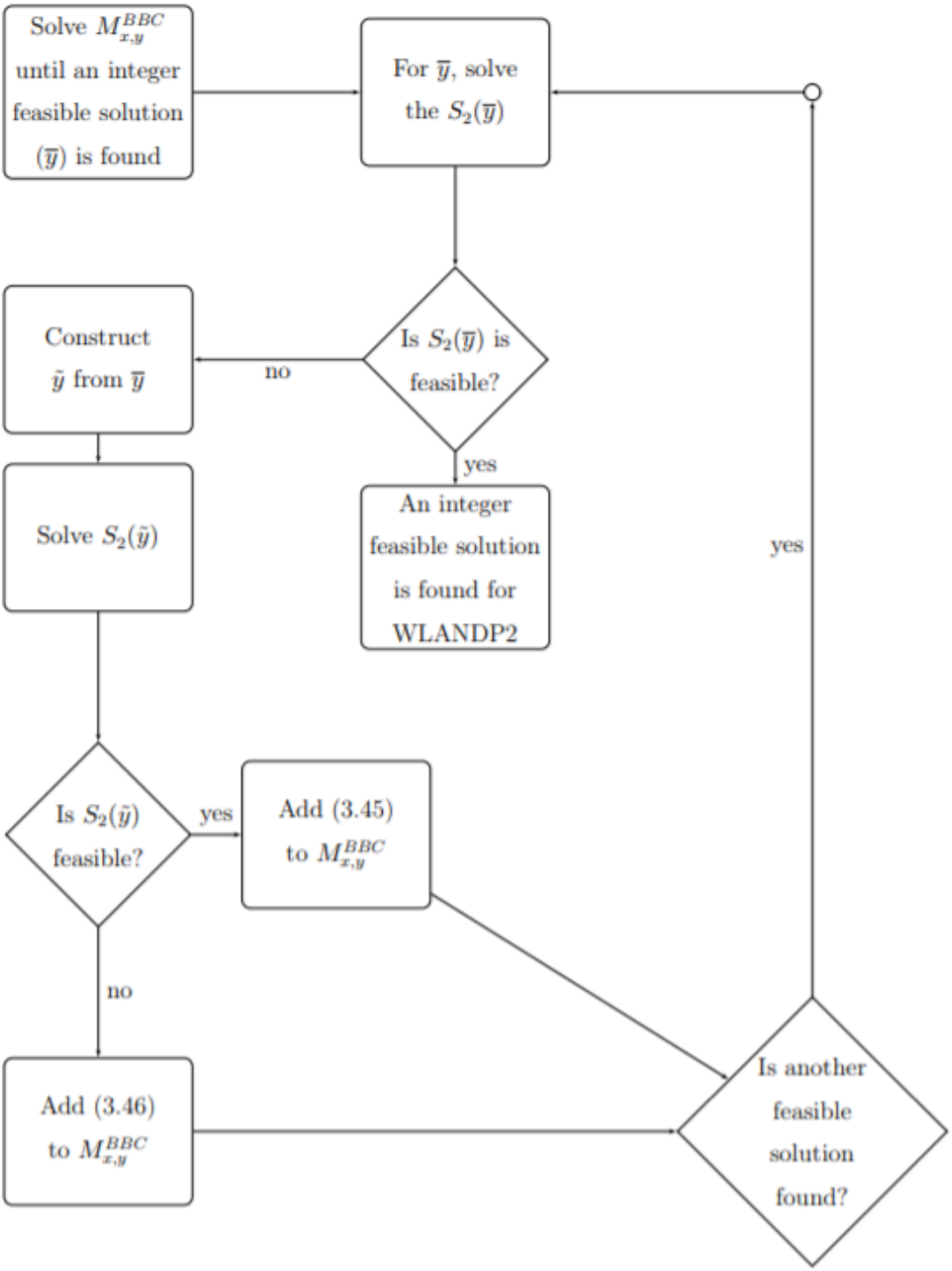


Figure 3.2: Flowchart of Branch-and-Benders cut (BBC) for WLANDP2

Table 3.4: Computational Results for MISOCP and BDTA1 for WLANDP1

					MISOCP				**BDTA1**						
Comb	$\|I\|$	$\|J\|$	w_i	r_{max}	**CPU (sec)**	**Nodes**	gap_1**(%)**	**Opt**	**CPU (sec)**	**Nodes**	gap_2**(%)**	**Opt**	**Aps**	**Cuts**	**feas (%)**
1				20,000	**114.8**	2,157	0	5	200.7	448	0	5	2	795	79.0
2	200	10	(100;150)	12,000	**1,438.4**	146,905	2.4	4	1,573.5	2,771	0	5	3	1,494	84.8
3				8,000	**2,566.0**	183,652	2.7	3	3,601.7	3,704	1.7	0	4	2,329	94.8
4				20,000	**1,315.4**	112,548	0	5	2,457.1	3,307	1.3	2	3	1,998	86.8
5	200	10	(200;250)	12,000	**1,883.2**	216,637	0	5	3,601.4	4,119	3.9	0	4	2,387	92.2
6				8,000	**2,989.2**	315,518	0.6	4	3,606.8	1,152	0.1	0	6	2,251	99.2
7				20,000	**196.3**	960	0	5	939.5	1,606	0.01	4	2	819	78.8
8	200	15	(100;150)	12,000	**3,535.6**	141,364	8.7	1	3,601.3	43,883	4.4	0	3	1,992	92.4
9				8,000	**2,845.0**	143,081	5.6	4	3,603.8	2,454	2.6	0	4	1,909	97.8
10				20,000	**2,114.9**	69,967	1.3	4	3,601.9	3,413	2.4	0	3	1,921	85.6
11	200	15	(200;250)	12,000	**343.9**	22,824	0	5	3,603.7	2,319	10.7	0	4	1,941	96.8
12				8,000	**2,983.0**	175,697	1.3	4	3,600.8	1,114	72.01	0	6	2,001	99.3
13				20,000	**1.1**	0	0	5	3.7	35	0	5	1	97	59.6
14	150	10	(100;150)	12,000	**46.4**	908,4	0	5	381.5	533	0	5	2	1,240	70.8
15				8,000	**1,009.8**	32,883	1.7	4	2,444.8	3,873	0	4	3	2,719	80.2
16				20,000	**26.9**	1,562	0	5	911.6	1,941	0	5	2	1,548	75.0
17	150	10	(200;250)	12,000	**107.4**	15,749	0	5	3,350.3	5,326	0.7	1	3	2,528	81.6
18				8,000	**2,650.7**	186,273	2.1	2	3,600.4	2,123	-0.47	0	5	2,288	96.2
19				20,000	**2.1**	0	0	5	11	113	0	5	1	167	60.1
20	150	15	(100;150)	12,000	**48.1**	290	0	5	619.3	1,269	0	5	2	956	76.8
21				8,000	**2,122.2**	92,387	2.0	3	2,441.7	3,059	1.8	3	3	1,838	92.2
22				20,000	**28.5**	698	0	5	773.8	2,676	0.03	4	2	680	75.0
23	150	15	(200;250)	12,000	**844.4**	43,863	0.3	4	3,223.6	4,724	0.9	2	3	2,120	86.9
24				8,000	**3,601.0**	240,979	2.1	0	3,647.9	2,066	15.8	0	5	1,758	96.6
				Opt				**97**				55			
				Average	**1,367.3**				2,517.6						

3.4.2 Comparison of MISOCP, BDTA2 and BBC for WLANDP2

All instances are the same as the instances solved for WLANDP1, but we now assume that $\mu = 0$. The computational results of MISOCP, BDTA2, and BBC for WLANDP2 are given in Table 3.5. Except the 'cuts' columns, the definitions of the remaining columns are the same as those in Table 3.4. In Table 3.5, the 'Cuts' column gives the average number of feasibility cuts added to master problem. If we have some instances whose solution status is 'OptTol', we include these instances to calculate the total number of instances solved to optimum.

When we compare the average CPU time over all instances, MISOCP outperforms the Benders decomposition type methods: BDTA2 and BBC. The average required time for MISOCP is 780.2, while the computational time for ILS2 is 1,218.5 and BBC requires 865.04 CPU seconds on the average. For each treatment combination, the best CPU time is given in bold. Instances become more difficult when demand is increased or r_{max} value is decreased. One important result that can be inferred from Table 3.4 is that the average CPU time in MISOCP is smaller than the one in BDTAs in difficult instances, where the demand of user terminals is high and r_{max} value is small. For these instances, the performance of BBC and BDTA2 algorithms gets worse. For example, in treatment combination 6 and 12, no instance could be solved to optimum by using BDTA2 and BBC while these instances could be solved optimally through MISOCP.

When compared to the methods in terms of the number of instances solved to optimum, BBC solves two more instances than MISOCP. We can conclude that the performance of BBC and MISOCP is similar in terms of the number of optimally solved instances.

In addition to the feasibility cut generation procedure in WLANDP2, optimality cuts must be generated in WLANDP1. This results in longer computation time compared to WLANDP2, which is a special case of WLANDP1. Therefore, the number of instances solved to optimum by BDTA is higher in WLANDP2 compared to WLANDP1.

When r_{max} value decreases, the average computation time for BDTA2 increases sig-

nificantly. The highest computational time occurs in the treatment combination where r_{max} value is 8000. When r_{max} value decreases, the number of instances solved to optimum does not increase. Similarly, in BBC algorithm, average CPU time increases with respect to r_{max} value except in the treatment combinations 9 and 24 in Table 3.5.

When compared the computational performance of the algorithms (BDTA2 and BBC), we show that BBC solves more instances to optimum with small CPU time. The reason is that BBC requires less number of feasibility cuts. It is an expected result since a more advanced procedure is used to generate feasibility cuts in BBC.

In Table 3.5 , the number of APs is high when r_{max} is small and demand value of user terminals are high. This is a similar result that we observed in Table 3.4 for WLANDP1. The reason is also the same, which we mentioned before. When the problem includes tighter capacity constraints, the number of APs in the optimal solution is high. Tighter capacity constraints can be obtained with low level of r_{max} and high level of demands of user terminals.

Table 3.5: Computational Results for MISOCP, BDTA2 and BBC for WLANDP2

	MISOCP				BDTA2					BBC			
Comb	**CPU**	**Nodes**	**gap (%)**	**Opt**	**CPU**	**Nodes**	**Opt**	**APs**	**Cuts**	**CPU**	**Nodes**	**Opt**	**Cuts**
1	25.8	69	0	5	**16.4**	33	5	2	248	31.6	68	5	218
2	2,190.7	166,471	6.8	2	**272.7**	293	5	3	899	290.5	793	5	381
3	1,143	60,071	3.3	4	1,769.5	1,466	4	4	1,814	**893.2**	2,954	5	1,690
4	909.3	72,939	0.8	4	**184**	339	5	3	635	360.7	760	5	495
5	**31.1**	604	0.02	5	1,624.3	1,373	4	4	1,812	1,153.4	7,126	5	1,988
6	**311,1**	5,858	0.1	5	3,600.1	1,236	0	6	2,819	3,604.9	36,746	0	1,525
7	91.7	827	0	5	**66.8**	0	5	2	405	685.7	9,431	5	433
8	2,181	73,922	0	3	**1,047.7**	0	4	3	1,085	2,475.9	29,207	3	857
9	1,548.7	33,138	8.6	3	3,601.3	0	0	4	2,170	**603.2**	18,613	2	496
10	1,482.4	49,001	4.4	4	800.5	810	5	3	916	**685.7**	9,432	5	433
11	**205.8**	5,689	0.02	5	3,601.3	2,257	0	4	2,231	2,475.9	29,208	3	857
12	**910.1**	15,078	0.04	5	3,601.7	81	0	6	840	3,600.1	6,500	0	1,130
13	1.5	0	0	5	**0.4**	0	5	1	18	0.6	1	5	6
14	16.2	178.8	0	5	**14.5**	38.8	5	2	85	35.3	76	5	78
15	935.5	83,380.2	4	4	**107.7**	447.6	5	3	513	138.2	2,190	5	159
16	**11.1**	70	0	5	14.4	41	5	2	85.2	16.6	63	5	50
17	734.7	41,954	0.9	4	**335.3**	363	5	3	1,295	375.2	86.3	5	729
18	2,171.4	92,269	6.7	2	3601.4	1,848	0	5	2,491	**1,987.2**	16,368	4	1,446
19	1.9	0	0	5	**0.4**	0	5	1	6.8	1.1	1	5	76
20	**27.8**	133	0	5	54.2	90	5	2	221	98.8	102	5	106.8
21	1,510.2	69,517	7.6	3	568	985.4	5	3	1,127.2	**231.6**	1,655	5	331.2
22	**77.4**	1,405	0	5	217.4	518	5	2	596	121.2	1,196	5	292.4
23	739,1	30,470	1	4	**542.3**	735	5	3	826	618.7	2,173	5	624.8
24	1,468	37,691	15	3	3,601.5	1,838	0	5	2,528	**275.6**	3,086	5	440
Opt				100			87					**102**	
Av.	**780.2**				1,218.5				1,069.4	865.04			**618.4**

3.4.3 The effects of Lift and Project Cuts(LPC) for Solving MISOCPs

When solving MISOCP for both versions of the problem, we set up IBM ILOG CPLEX to add LPCs aggressively by setting LiftProj parameter of IBM ILOG CPLEX to 2. Adding these cuts to mixed integer convex programs is a relatively new issue. The cuts generated in MILPs are extended to the mixed integer nonlinear programs. In Table 3.6, we give solution times, number of open nodes in the ($B\&B$) tree, gap and the number of instances solved to optimum in each five replications with aggressive use of LPCs and default use of the same cuts. We give the smaller CPU time in bold in Table 3.6. We refer the reader to [14] for the details about the effects of LPCs in mixed integer convex programs.

3.5 Conclusion

In this chapter, we considered a nonlinear facility location problem, which occurred in wireless local area network design. The problem has two versions depending on the objective functions. We develop two exact solution procedures (BDTA1 and MISOCP) for WLANDP1 and three exact methods (BDTA2, MISOCP, and BBC) for WLANDP2. We evaluate the performance of the methods on randomly generated instances.

To the best of our knowledge, MISOCP is the first formulation that can be solved by commercial solvers. This reformulation is easy to implement compared to Benders type algorithms. For WLANDP1, in all settings BDTA1 requires more computational time than MISOCP. We can conclude that MISOCP has better computational performance. For WLANDP2, we cannot mention a method to be significantly superior to other methods. As long as the objective function is convex, the MISOCP formulation can be applicable for stochastic version of the problem, where uncertain parameter values are generated from a discrete number of scenarios.

In addition to MISOCP reformulation for WLANDP1 and WLANDP2, we propose a Benders type decomposition algorithms for two variants of the problem in order to handle nonlinearities in the model. This decomposition frame is very similar to

Integer L-shaped algorithm that is widely used for two-stage stochastic integer problems. The main advantage of this algorithm is that it can be used for both variants of WLANDP independent of the objective function.

BD algorithm, which is an iterative approach ([40]) and implemented for WLANDP2, is not applicable for the more general variant, WLANDP1. The reason is that the subproblem should be a feasibility problem to implement BD algorithm. Since the objective function of WLANDP1 includes two terms, implementation of BD given in [40] is not possible. However, our proposed BDTA can be implemented for both variants. Note that, both feasibility and optimality cuts are generated in BDTA1 while BDTA2 requires only feasibility cuts.

In recent years, by taking the advantage of *callback functions* in commercial solvers, implementing Benders decomposition type algorithms in a single branch-and-bound tree becomes easier. In this study, we also use *lazyconstraint callback* function in IBM ILOG CPLEX so that we could add feasibility or optimality cuts when they are necessary by interrupting the branch-and-bound tree of the master problem.

In the computational study, we observe that the performance of Benders type algorithms becomes worse for the instances with small r_{max} value. By using Benders type decomposition algorithms, optimal solutions can be found for easy instances, where the number of open APs is small, in reasonable CPU time. However, for difficult instances in which high number of APs needs to be opened, the performance of MISOCP is better than the one of Benders type decomposition algorithms.

In the decomposition frame for the WLANDP1 and WLANDP2, the lazyconstraint callback function is invoked when an integer feasible solution from MP is found. However, it is also possible that adding optimality cuts to the MP when y values are not integer. Implementing such a decomposition algorithm could be a future research.

We also observe that for some instances, the performance of BBC algorithm is better than the performance of MISOCP. In order to improve the performance of MISOCP, we can implement a decomposition algorithm for this reformulation. In this decomposition, MP includes the variables x_{ij} and y_{jk} and the subproblem will be just a feasibility problem where the conic constraints' feasibility are checked. In this case,

the decomposition will include only feasibility cuts since there is no objective function in the subproblem.

Table 3.6: Comparison results obtained with/without lift-and-project cuts on MISOCP for WLANDP1

	MISOCP (with parameter 2)				**MISOCP (default)**			
Pr	**CPU (sec)**	**nodes**	**gap**	**Opt**	**CPU (sec)**	**nodes**	**gap**	**Opt**
1	**114.8**	2157.6	0	5	159.1	7771.2	0	5
2	**1438.4**	146905.2	2.37	4	2688.1	203281.2	8.8	2
3	**2498.6**	235550.2	3.49	4	3600.1	188324.8	12.1	0
4	**1313.5**	112548.4	0	5	3545.4	205317.4	5.95	1
5	**1883.2**	216637.6	0	5	2525.9	147051.6	1.31	3
6	**2989.2**	315518.8	0.61	4	3532.3	251868	9.53	1
7	196.3	960.2	0	5	**177.6**	2127.2	0	5
8	**3535.7**	141364.8	8.67	1	3600.2	120062	13.13	0
9	**2844.9**	143081.6	5.6	4	3600.2	94608.2	28.8	0
10	**2114.8**	69967.8	0	4	3600.2	105296	9.25	0
11	**343.9**	22824.4	0	5	3600.2	82160.2	10.34	0
12	**2983.1**	175697.6	1.94	4	3600.5	102187.8	17.63	0
13	1.1	0	0	5	**0.85**	0	0	5
14	**46.5**	908.4	0	5	60.2	3160	0	5
15	**1009.8**	32882.6	1.7	4	3241.8	283841	9.54	1
16	**26.9**	1562.4	0	5	68.5	4887	0	5
17	**107.5**	15749	0	5	2513.6	241234.2	3.51	2
18	**2650.7**	148966	2.53	2	3600.1	263259.4	10.98	0
19	2.1	0	0	5	**1.8**	0	0	5
20	**48.1**	290.6	0	5	102.1	3971.4	0	5
21	**2122.2**	92387.4	2	3	3600.4	192701	12.36	0
22	**28.5**	698.4	0	5	84.7	2588.4	1.2	5
23	**844.4**	43863.6	0.29	4	2452.2	77986.8	3.63	2
24	3601.0	240979.4	2.1	0	3600.1	174134.2	21.13	0
Total				**98**				52
Average	2239.4				2231.5			

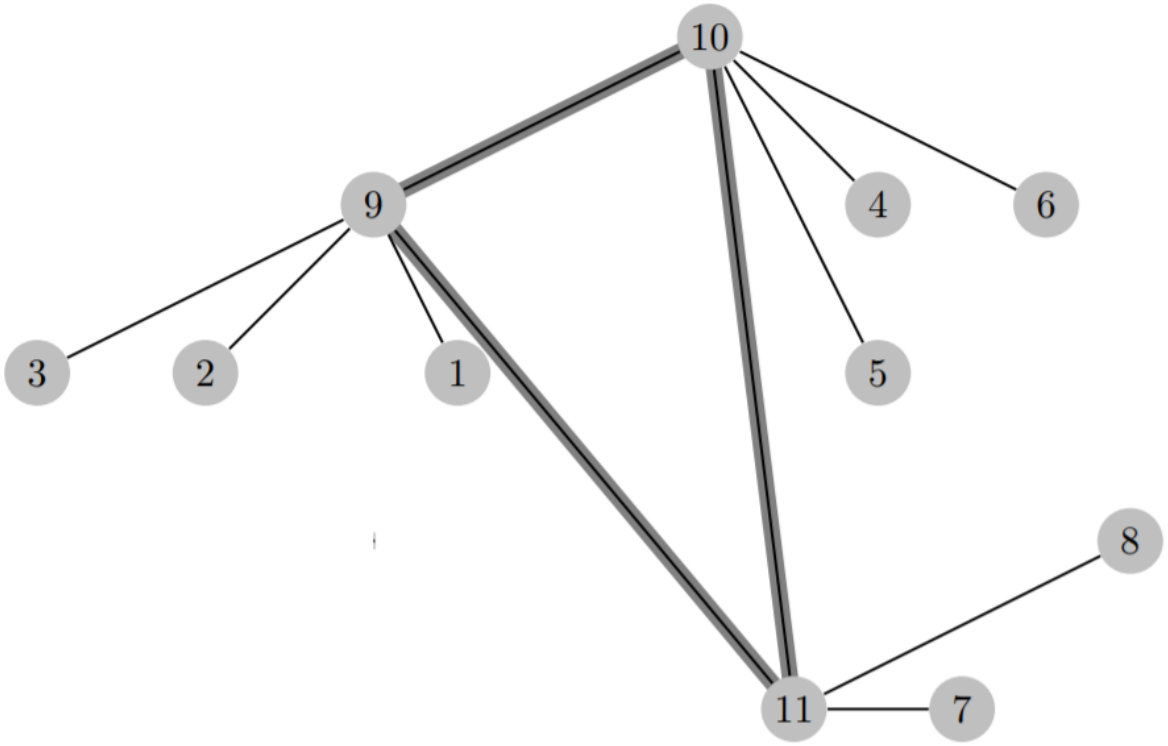
CHAPTER 4

QUADRATIC CAPACITATED CONCENTRATOR LOCATION PROBLEM WITH SINGLE ASSIGNMENT *(QCCLP)*

In this chapter, we study a quadratic capacitated concentrator location problem, which arises in telecommunication network systems. The aim is to determine the locations of the concentrators (*hubs*) and assignments of non-hub nodes (*terminals*) to hub nodes under cost minimization objective. The objective function consists of two cost terms: the *fixed cost of opening hubs (concentrators)* and the *routing cost*. The problem was first studied by [53]. For QCCLP, we propose an exact Benders type decomposition algorithm, which we call the *branch-and-check algorithm*. In Section 4.1, we give the problem definition and mathematical formulation for *QCCLP*. Then, we explain our proposed decomposition algorithm and discuss some enhancement steps in Section 4.2. Next, in Section 4.3, we provide computational results. Then, we give the computational results for comparing branch-and-check algorithm and automatic Benders decomposition in Section 4.4. Finally, we conclude the chapter and give future research directions in Section 4.5.

4.1 Problem Definition and Mathematical Formulation

Figure 4.1: An example network with access and backbone networks for QCCLP



In telecommunication network systems, a large number of terminals (user points) exists. Terminals can be seen as users which have traffic demand. In these systems, there are electronic devices that are called as concentrators, hubs, or multiplexers. As the system has many users, roles of concentrators are to compress traffic, combine multiple signals from different users (terminals), and forward combined demand to the relevant destination points.

In order to meet the demand of a terminal, each terminal is assigned to a concentrator. In some telecommunication network systems, concentrators can also be connected to a central node. On the other hand, there can be direct links between any concentrator pair which results in a complete network. Constructing such a system where all concentrators are connected with each other by direct link could be costly. To overcome this issue, concentrators can be connected to a single central node by a ring.

Due to the existence of the concentrators in the system, telecommunication networks generally have multi-layer structure. The network between the terminals (non-hub nodes) and concentrators can be seen as the lower level of this structure. Aggregated traffic at the lower level is send to the upper level of the system. As mentioned in [78], telecommunication networks have many levels. In general, lower level network is less dense than the upper level [78].

The problem that we considered in this chapter includes two levels. These levels are backbone network and access network. Both backbone and access network can have several structures: *ring, complete, star, tree, etc*. The structure in QCCLP is star access network and complete backbone traffic.

We give an example network representing the environment of the QCCLP. In Figure 4.1, Nodes 9, 10, and 11 are hub nodes and the remaining nodes are non-hub nodes. Bold arcs are *backbone arcs*, while the arcs between non-hub nodes and hub nodes are called *access arcs*. One of the assumptions of the QCCLP is that at least one hub location must be visited to satisfy the demand of each origin-destination pair.

In Figure 4.1, if there exists positive demand between the nodes (terminals) 1 and 2, the concentrator location 9 must be visited. The path of the flow is *Node 1- Node 9- Node 2* to send the flow from Node 1 to Node 2. This amount of flow does not

contribute to backbone traffic since both nodes (1 and 2) are assigned to the same concentrator.

When two non-hub nodes are assigned to different hub nodes (e.g., nodes 1 and 5), the path of the flow from Node 1 to Node 5 is *Node 1-Node 9-Node 10- Node 5*. In other words, no direct shipment between two terminal nodes is allowed in QCCLP. In this example, two concentrators must be used to send the flow between two different non-hub nodes.

We are given a set of nodes I. Let t_{im} be the demand or flow to be sent from node i to node m. There is a corresponding fixed cost f_i incurred if node i is selected as a hub. d_{ij} represents the distance between nodes i and j. In QCCLP, each hub has the same capacity M. In the objective function (4.1), α and β represent collection and distribution unit parameters, respectively.

QCCLP can be defined as follows. A subset of node set I is chosen as hub locations and the remaining nodes are assigned to these hub locations. The objective to minimize is the sum of fixed cost of opening hubs and routing cost between non-hub nodes and hub nodes. To transfer flow from node i to node m, at least one hub location must be visited. As each non-hub node (terminal) must be assigned to exactly one concentrator, QCCLP is in the class of single allocation hub location problems (SAHLP).

x_{ij} represents the 0-1 decision to assign node i to node j. If node j is selected as a hub, then $x_{jj} = 1$, otherwise $x_{jj} = 0$. By using these parameters and the decision variables, the mathematical model given by [53] is below:

$$\min \sum_{j\in I} f_j x_{jj} + \sum_{i\in I}\sum_{j\in I:i\neq j} d_{ij}x_{ij}\left(\sum_{m\in I}\alpha t_{im} + \beta t_{mi}\right) \tag{4.1}$$

(QCCLP) s.t.
$$\sum_{j\in J} x_{ij} = 1, \qquad \forall i \in I. \tag{4.2}$$

$$x_{ij} \leq x_{jj}, \qquad \forall i,\ j \in I, i \neq j. \tag{4.3}$$

$$\sum_{i\in I}\sum_{m\in I}(t_{im}+t_{mi})x_{ij} + \sum_{i\in I}\sum_{m\in I}(t_{im}+t_{mi})x_{ij}(1-x_{mj}) \leq Mx_{jj}, \forall j \in I. \tag{4.4}$$

$$x_{ij} \in \{0,1\} \qquad \forall i, \quad j \in I. \tag{4.5}$$

In this formulation, the objective function (4.1) minimizes total cost of opening hubs and routing cost of assigning non-hub nodes to hub nodes. The second term in the objective function $\sum_{i\in I}\sum_{j\in I:i\neq j} d_{ij}x_{ij}(\sum_{m\in I}\alpha t_{im})$ is the total cost related to collection of data. Lastly, the term $\sum_{i\in I}\sum_{j\in I:i\neq j} d_{ij}x_{ij}(\sum_{m\in I}\beta t_{mi})$ is the total cost related to distribution of data.

Constraints (4.2) guarantee that each node is assigned to exactly one hub. Constraints (4.3) ensure that node i can be assigned to node j only if j is a hub. Constraints (4.4) ensure that total flow that uses the capacity of hub j cannot exceed the capacity M. In Constraints (4.4), $\sum_{i\in I}\sum_{m\in I}(t_{im}+t_{mi})x_{ij}$ is the total flow of nodes that are assigned to hub j and $\sum_{i\in I}\sum_{m\in I}(t_{im}+t_{mi})x_{ij}(1-x_{mj})$ is the flow occurred on backbone links. Constraints (4.5) are the sign restrictions on the decision variables. We will give the branch and check algorithm for this nonlinear integer formulation of QCCLP in Section 4.2.

4.2 A Branch and Check Algorithm for QCCLP

We propose an exact branch-and-check algorithm, which is an implementation of the logic based Benders decomposition in a single branch-and-cut tree of the master problem. We propose different variants of this algorithm by making some enhancements on the master problem and cut generation phases.

A single allocation linear capacitated hub location problem, where the capacity con-

straints are linear, is NP-hard and QCCLP is a general variant of SAHLP by replacing the linear constraints with nonlinear constraints. Therefore, QCCLP is an NP-hard problem. One source of difficulty in solving this problem is the nonlinearity in the capacity constraints. Also, the problem has a combinatorial nature which is another source of difficulty. In QCCLP, the capacity constraints (4.4) include multiplication of binary variables.

The feasibility check of each hub and adding necessary feasibility cuts can be done in the *IloLazyconstraint callback function* of CPLEX. Using lazy constraint function in the implementation of Benders decomposition can also be seen in several studies (e.g., [35], [60]).

4.2.1 Master Problem & Subproblem

For the *QCCLP*, we remove the Constraints (4.4) and obtain an uncapacitated single allocation facility location model as *MP* given below:

$$\min \sum_{j \in I} f_j x_{jj} + \sum_{i \in I} \sum_{j \in I: i \neq j} d_{ij} x_{ij} \left(\sum_{m \in I} (\alpha t_{im} + \beta t_{mi}) \right)$$

(MP) s.t.

$$(4.2), (4.3), (4.5)$$

$$\sum_{i \in I} \mu_{l_i} x_i \geq \mu_{l_0} \qquad \forall l \in \omega_{feas} \tag{4.6}$$

The master problem is NP-hard. In MP of this decomposition, Constraint (4.6) represent the feasibility cuts. Given a feasible or an integer solution to the MP ($\overline{x}$), in the subproblem phase, we check whether Constraints (4.4) are satisfied or not. Therefore, the subproblem phase is just a feasibility check. If a solution ($\overline{x}$) to MP turns out to be infeasible for the original problem *(QCCLP)*, then we add feasibility cuts to the MP. Therefore, ($\overline{x}$) can be eliminated from the solution space of MP. Such kind of cuts are also used in logic based Benders decomposition (e.g., [45]) and combinatorial Benders cut (e.g., [20]).

In the case of infeasibility, we know that at least one of the x_{ij} variables must take a value different than the current solution ($\overline{x}$). To satisfy this, we use three variants

of feasibility cuts (i) *nogoods cuts*, (ii) *multiple feasibility cuts*, and (iii) *extended feasibility cuts*.

Our overall Branch & Check algorithm is given in Algorithm 1. Moreover, in order to make the algorithm clear for the readers, we give a flow chart of the algorithm in Figure 4.2.

Algorithm 1 Branch and Check Algorithm for *QCCLP*

1: Solve the *MP* and find an integer feasible solution $(\overline{x})$
2: *For each hub in opened location j in solution $\overline{x}$*:
3: **if** $\sum_{i\in I}\sum_{m\in I}(t_{im}+t_{mi})\overline{x}_{ij}+\sum_{i\in I}\sum_{m\in I}(t_{im}+t_{mi})\overline{x}_{ij}(1-\overline{x}_{mj}) \leq M\overline{x}_{jj}$ **then**
4: **if** The current solution is the best solution so far for the QCCLP **then**
5: *Update the incumbent solution of MP*
6: **end if**
7: **else**
8: Add one of the feasibility cuts given in Inequality (4.11) and Inequality (4.14) to the *MP*.
9: **end if**

After finding the integer feasible solution $(\overline{x})$ to the MP, the nodes that are selected as concentrators and the assignments of non-hub nodes to these concentrators are known. However, MP includes the relaxation of the nonlinear constraints. Therefore, the solution $(\overline{x})$ may not satisfy the hub capacities. To check the hubs whose capacity is exceeded by assigned demand, we evaluate Inequality (4.7) by using $\overline{x}$ for each open hub j.

$$\sum_{i\in I}\sum_{m\in I}(t_{im}+t_{mi})\overline{x}_{ij}+\sum_{i\in I}\sum_{m\in I}(t_{im}+t_{mi})\overline{x}_{ij}(1-\overline{x}_{mj}) \leq M\overline{x}_{jj} \tag{4.7}$$

In this inequality, $\sum_{i\in I}\sum_{m\in I}(t_{im}+t_{mi})\overline{x}_{ij}$ gives the total amount of assigned demand to the hub location j in solution $\overline{x}$. The term, $\sum_{i\in I}\sum_{m\in I}(t_{im}+t_{mi})\overline{x}_{ij}(1-\overline{x}_{mj})$ represents the current backbone traffic between hub location j and other hubs in $\overline{x}$.

After evaluating Inequality (4.7) for each open hub j, we have two cases:

- *Case 1:* When Inequality (4.7) is satisfied for each open hub location j, an integer feasible solution for QCCLP is found. This corresponds to the Step 3

of Algorithm 1.

- *Case 2:* If there exists a hub j which does not satisfy Inequality (4.7), then the solution $\overline{x}$ is not a feasible solution for QCCLP. In this case, one of the possible feasibility cuts (*multiple feasibility cut* and *extended feasibility cut*) is added to the current node of MP.

Evaluating whether the capacity of the hubs is satisfied or not, and adding feasibility cuts to the MP constitute the subproblem phase in the branch-and-check algorithm. Note that, the subproblem phase does not have any model including decision variables.

Master problem *(MP)* finds the assignments of non-hub nodes to hub nodes and selections of hubs among all nodes by only considering hub opening cost and routing cost between non-hub nodes to hub nodes. In order to obtain better solutions in master problem stage, we add valid inequalities to the *MP* at the beginning. The detailed discussion on valid inequalities for the master problem is given in Section 4.2.2.

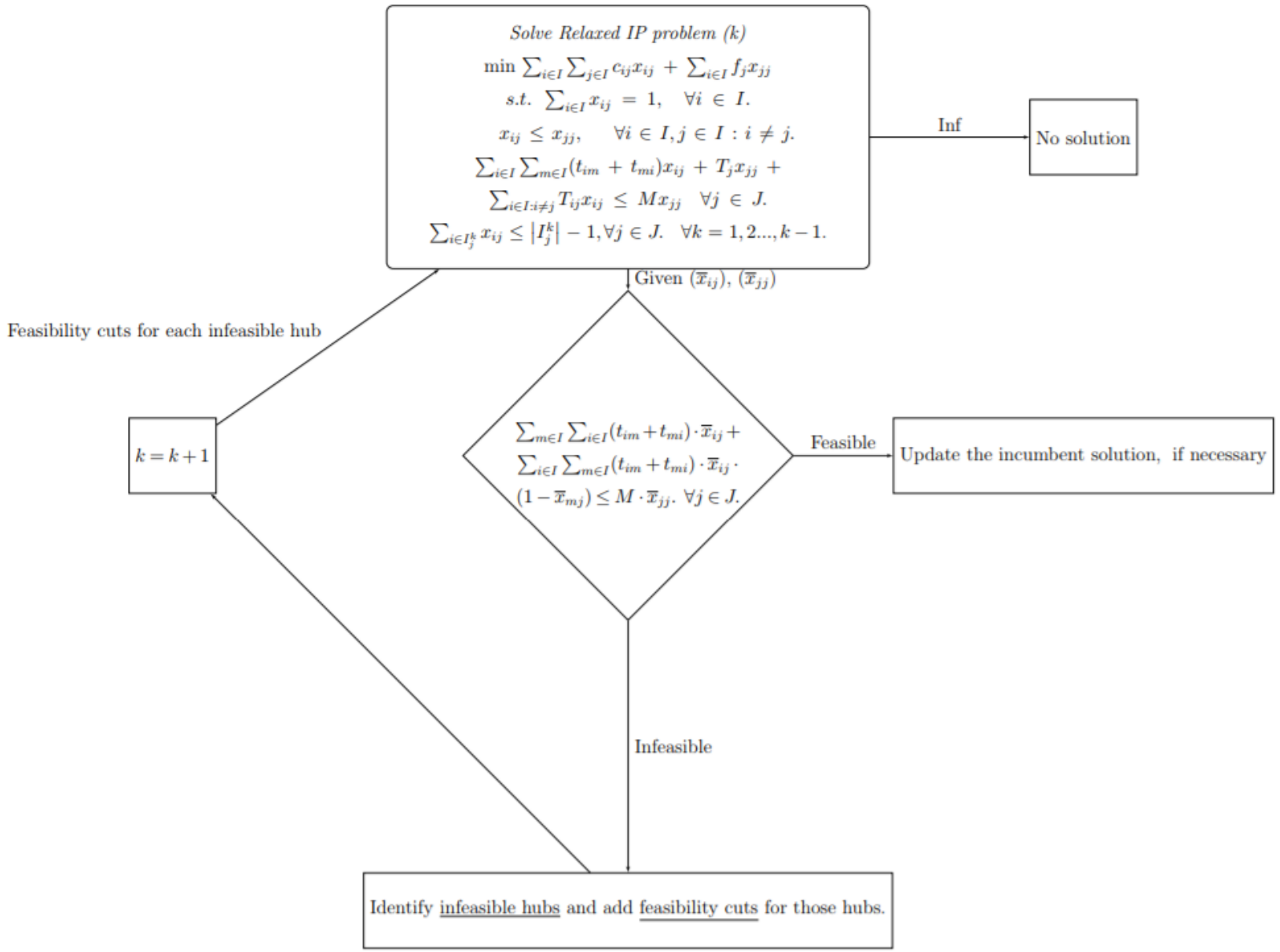
Figure 4.2 demonstrates the flow of the branch-and-check algorithm that is applied for QCCLP. As can be seen from this figure, the algorithm starts with obtaining an integer feasible solution to the relaxed IP problem which is MP in the branch-and-check (B&Ch). In addition to MP that we explained in Section 4.2.1, MP includes relaxed version of nonlinear capacity constraints. These inequalities include two main parts: *total amount of assigned demand to the hub* j and *estimated backbone traffic* between concentrators.

If the MP does not give a feasible solution, this means that QCCLP is infeasible. When an integer feasible solution is found in the MP phase, the capacity constraints of the hubs are evaluated (*the decision node in Figure 4.2*). After checking the feasibility of hub capacities, there are two options in this figure. When the algorithm proceeds to the *'feasible'* side, it implies that we might have found an incumbent solution. If the status of the decision node in the figure is infeasible, the hubs whose capacity constraints are not satisfied by the master problem solution $\overline{x}$ are found. For these hubs, feasibility cuts are added to the MP.

After finding the feasible solution to MP, all remaining steps given in Figure 4.2 are

made in the branch-and-bound node of MP. To interrupt the branch-and-and bound tree of the MP, we use lazycallback function that is available in commercial solvers (e.g., IBM ILOG CPLEX). In branch-and-check algorithm, the MP is solved only once. Therefore, it has similarities with the general branch-and-cut algorithm. In Figure 4.2, k represents the counter of lazycallback functions invoked. In each lazy-callback function, we add multiple feasibility cuts since the number of infeasible hubs is more than one. Moreover, the set of hubs at which capacity constraints are not satisfied may change. I_j^k represents the set of hubs whose capacity constraints are not met in lazycounter.

Figure 4.2: A Branch and Check Algorithm for QCCLP



4.2.2 Valid Inequalities for the Master Problem

In order to strengthen the formulation of MP, we use two variants of valid inequalities. To generate these inequalities, we use the algorithm which was referred to *traffic bound algorithm* in [53]. They propose an algorithm to find a lower bound on the optimal amount of backbone traffic on backbone links by solving knapsack problems. Steps of the algorithm are given in Algorithm 2.

By using the lower bound values obtained from Algorithm 2, we add Inequality (4.8) to the *MP* as valid inequalities. These valid inequalities are called as VI1 in this study.

$$VI1 : \sum_{i \in I} \sum_{m \in I} (t_{im} + t_{mi}) x_{ij} + T_j x_{jj} \leq M x_{jj} \quad \forall j \in I. \tag{4.8}$$

In Inequality (4.8), T_j represent the minimum amount of backbone traffic that would occur if node j is selected as a hub location. In each iteration of the traffic bound algorithm, the knapsack problem given in the third step of the Algorithm 2 is solved. As described by [53], a similar algorithm as Algorithm 2 can be used to find lower bound on backbone traffic (T_{ij}) that would occur if node i were assigned to node j.

Therefore, another valid inequality (VI2) that could be added to the MP is given below:

$$VI2 : \sum_{i \in I} \sum_{m \in I} (t_{im} + t_{mi}) x_{ij} + T_j x_{jj} + \sum_{i \in I : i \neq j} T_{ij} x_{ij} \leq M x_{jj} \quad \forall j \in I. \tag{4.9}$$

Generating the valid inequality given in (4.9) requires solving O(n^2+n) integer knapsack problems, where n represents the number of nodes in the network. Therefore, it might be costly to compute all T_{ij} values for large size instances.

In Algorithm 2, all decision variables in the knapsack problems are binary. In our computational study, we observe that solving the integer knapsack problems for large size instances may require too much computational time. In order to handle this drawback, we also solve the LP relaxations of the knapsack problems for finding valid lower bounds on the amount of backbone traffic. We observe that solving the knapsack problems by using an LP relaxation requires significantly smaller CPU times than solving them as IPs. On the other hand, the objective function values of the

knapsack problems in LP relaxation are smaller than the values in IPs. Therefore, there is a tradeoff between obtaining a strong lower bound on the backbone flow and the required time to obtain this bound.

Algorithm 2 Traffic Bound Algorithm for QCCLP

1: $counter \leftarrow 1$ and $d_i = \sum_{m \in I}(t_{im} + t_{mi})$ for each node i.

2: For each node i, solve the following *IP knapsack problem*.

3: Let a_i be $\sum_{m \in I} T_{im} = \sum_{m \in I}(t_{im} + t_{mi})$.

$$T_i = \min \sum_{m \in I: m \neq i} T_{im}(1 - u_m)$$

(Traffic Bound 1) s.t.

$$\sum_{m \in I: m \neq i} d_m u_m \leq M - d_i$$

$$u_m \in \{0, 1\} \qquad \forall m \ \in I \setminus \{i\}.$$

4: **if** $a_i + T_i > d_i$, **then**

5: $d_i \leftarrow (a_i + T_i)$ *&* $counter \leftarrow (counter + 1)$ *go to step (3) and solve the same* IP model *with updated* d_i *values.*

6: **else**

7: STOP, use T_i values for each node i to generate valid inequalities that are added to the master problem.

8: **end if**

In Inequality (4.9), T_{ij} values are calculated by solving the knapsack problems which are modified version of the model given in the third. step of Algorithm 2. The modified version of the knapsack problem to find a lower bound on the amount of backbone flow is below:

$$T_{ij} = \min \sum_{m \in I: m \neq i,j} T_{im} \cdot (1 - u_m)$$

(Traffic Bound 2) s.t.

$$\sum_{m \in I: m \neq i} d_m \cdot u_m \leq M - d_i - d_j$$

$$u_m \in \{0, 1\} \qquad \forall m \ \in I \setminus \{i, j\}.$$

As the capacity of the hub is used by both hub node i and non-hub node j, right hand

side of the capacity constraints is updated as $M - d_i - d_j$ in this formulation to find T_{ij} values.

4.2.3 Nogoods cuts

Assume that we have a solution vector $\overline{x}$ in any step of solving master problem. Then, set $S(\overline{x}) = \{(i,j) : \overline{x}_{ij} = 1\}$ and the complement of this set $\overline{S}(\overline{x}) = \{(i,j) : \overline{x}_{ij} = 0\}$.

By using these sets, the general feasibility cut or *nogood* cut in the literature of branch-and-check algorithm can be generated as:

$$\sum_{(i,j)\in S(\overline{x})} (1 - x_{ij}) + \sum_{(i,j)\in \overline{S}(\overline{x})} x_{ij} \geq 1. \tag{4.10}$$

This cut (4.10) is also known as *combinatorial cut* in combinatorial Benders decomposition ([20]). The main disadvantage of *"no-good cuts"* is that they are generally weak. This type of cuts only cut off the current solution from the region. Therefore, the number of iterations required for convergence is usually high. However, problem specific cuts can remove the points which are similar with the current infeasible solution.

4.2.4 Multiple feasibility cuts (MFC)

If there are hubs whose capacities are not satisfied by the solution ($\overline{x}$), we also propose a stronger cut, which is problem specific, than no-good cuts. In no-goods cuts, we add only one cut to exclude the infeasible solution. In order to detect the source of infeasibility in the solution $\overline{x}$ in more detail, we generate a feasibility cut for each hub whose capacity is not satisfied. Therefore, we call feasibility cuts as *multiple feasibility cut (MFC)*.

In the representation of the cut, k represents the counter for *lazyconstraintcallback* function used in the algorithm.

I_j^k : the set of nodes assigned to hub j in *lazycounter* k.

I_{inf}^k : the set of hubs whose capacity constraints are not satisfied by MP solution $\overline{x}$ in *lazycounter* iteration k.

Throughout the study, we call the hubs whose capacity constraints are not satisfied by the current MP solution as *infeasible hubs*.

$$\sum_{i \in I_j^k} x_{ij} \leq \left|I_j^k\right| - 1, \forall j \in I_{inf}^k \quad \forall k = 1, 2..., k-1. \tag{4.11}$$

When $\overline{x}$ gives an infeasible solution in the subproblem phase, at least one of the nodes that are assigned to an infeasible hub must be removed. By using this inequality (4.11), we satisfy this requirement for each hub whose capacity is exceeded. To clarify the MFCs and the difference between no-goods cuts, we give the following example.

Example 4.2.1. *Assume a network with 7 nodes and in the MP's solution, we have the following assignments:* $x_{11} = x_{31} = x_{41} = 1$ *and* $x_{22} = x_{52} = 1$. *In this example, nodes 1 and 2 are the hubs. We will explain the multiple feasibility cut and no-good cut under the assumption that both hubs are infeasible. In that case, no-good cut which is given in Inequality (4.10) is below:*

$$(1-x_{11})+(1-x_{22})+(1-x_{31})+(1-x_{41})+(1-x_{52})+x_{12}+x_{13}+x_{14}+x_{15}+x_{21}+x_{23}+x_{24}+x_{25}+$$

$$x_{32} + x_{33} + x_{34} + x_{35} + x_{42} + x_{43} + x_{44} + x_{45} + x_{51} + x_{53} + x_{54} + x_{55} \geq 1.$$

To find a better cut, we can use a minimal subset of items which still returns an infeasible solution. In MFC, we obtain the following cuts, as there are two infeasible hubs in this example.

- *For hub location 1;*

$$(1 - x_{11}) + (1 - x_{31}) + (1 - x_{41}) \geq 1.$$

- *For hub location 2;*

$$(1 - x_{22}) + (1 - x_{52}) \geq 1.$$

The no-good cut includes 25 terms, while the MFCs include at most 5 terms. As the number of items in MFCs are less than the number of items in no-good cuts, MFCs are stronger than no-good cuts. In our computational study, we observe that computational time required to solve the MP with high number of feasibility cuts is not huge.

Proposition 4.2.2. *For a given integer solution $\overline{x}$ to* MP*, if constraint (4.4) is not satisfied for location j, then*

$$\sum_{i \in I : \overline{x}_{ij}=1} (1 - x_{ij}) \geq 1. \tag{4.12}$$

is a feasibility cut for MP.

Proof. The key idea behind this feasibility cut is that we cannot obtain a feasible solution satisfying capacity constraints without removing at least one node from hub location j. In order to prove this observation, we need to show that if we assign an additional node to the infeasible hub, total amount of change on the flow of the hub is positive. In this proof, we will use the following notation:

- $\overline{j}$: a specific hub location whose capacity is exceeded by the flows that are assigned in solution $\overline{x}$.
- a_i : total inflow and outflow related to node i.
- $\overline{x}_{i\overline{j}}$: the solution values that are obtained from master problem. These values are used as parameters in the subproblem.
- z_{jl} : backbone flow from hub location j to hub location l.

Let $I_{\overline{j}}$ denote the set which includes the nodes that are assigned to $\overline{j}$. The non-hub nodes for this set can be represented as $i_1, i_2, ..., i_n$. As $\overline{j}$ is a hub, $\overline{j}$ is included in this set. Therefore, we have $\overline{I}_{\overline{j}}$= $\{i_1, i_2, \overline{j}, ...i_n\}$.

For $\overline{j}$, Inequality (4.13) holds.

$$\sum_{i \in I} a_i \cdot \overline{x}_{i\overline{j}} + \sum_{l \in I : l \neq \overline{j}} (z_{\overline{j}l} + z_{l\overline{j}}) > M. \tag{4.13}$$

In this inequality $\sum_{i \in I} a_i \cdot \overline{x}_{i\overline{j}}$ is the total demand that is assigned to hub location, $z_{\overline{j}l}$ denotes backbone traffic from hub $\overline{j}$ to hub l. Lastly, $z_{l\overline{j}}$ represents backbone traffic from hub l to hub $\overline{j}$.

In Inequality (4.13):

$$\sum_{i \in I} a_i \cdot \overline{x}_{i\overline{j}} = \sum_{i \in \overline{I}_{\overline{j}}} \sum_{m \in I : m \neq i} (t_{im} + t_{mi}).$$

Assume that we have k other hubs different than $\overline{j}$ in the network. In order to express total backbone traffic between the hubs $\overline{j}$ and the other hubs (l_1, l_2,..., l_k), we divide the calculations into two parts such as $z_{\overline{j}l_i}$ and $z_{l_i\overline{j}}$.

$$z_{\overline{j}l_1} + z_{\overline{j}l_2} + ... + z_{\overline{j}l_k} = \sum_{I \in I_{\overline{j}}} \sum_{m \in I \setminus I_{\overline{j}}} t_{im}.$$

$$z_{l_1\overline{j}} + z_{l_2\overline{j}} + ... + z_{l_k\overline{j}} = \sum_{i \in I \setminus I_{\overline{j}}} \sum_{m \in I_{\overline{j}}} t_{im} = \sum_{m \in I \setminus I_{\overline{j}}} \sum_{i \in I_{\overline{j}}} t_{mi}.$$

We can arbitrarily choose a node b which is a non-hub node not assigned to hub $\overline{j}$.

Since any node $\overline{i}$ is assigned to the hub $\overline{j}$, total inflow to the node b_1 and total outflow from that node b will use the capacity of $\overline{j}$. If the node b is assigned to the hub $\overline{j}$, the increase on the amount of traffic assigned will be:

$$\sum_{m \in I : m \neq b_1} (t_{bm} + t_{mb}).$$

As b is assigned to hub $\overline{j}$ from its previous set $I \setminus I_{\overline{j}}$, the amount of change on $z_{\overline{j}l}$ is $\sum_{m \in I \setminus I_{\overline{j}}} t_{bm} - \sum_{m \in I_{\overline{j}}} t_{mb}$ on $z_{\overline{j}l}$. Similarly, the amount of change on $z_{l\overline{j}}$ is $\sum_{m \in I \setminus I_{\overline{j}}} t_{mb} - \sum_{m \in I_{\overline{j}}} t_{bm}$.

Therefore, total change on the capacity usage of the hub $\overline{j}$ is as:

$$\cancel{\sum_{m \in I_{\overline{j}}} (t_{bm} + t_{mb})} + \sum_{m \in I \setminus I_{\overline{j}}} (t_{bm} + t_{mb}) + \sum_{m \in I \setminus I_{\overline{j}}} t_{bm} + \sum_{m \in I \setminus I_{\overline{j}}} t_{mb} - \cancel{\sum_{m \in I_{\overline{j}}} t_{mb}} - \cancel{\sum_{m \in I_{\overline{j}}} t_{bm}}$$

To sum up, *total change* is

$$2 \sum_{m \in I \setminus I_{\overline{j}}} (t_{bm} + t_{mb}).$$

Since demand between two nodes are positive values, total change is strictly greater than zero. This concludes that in order to satisfy capacity constraints for $\overline{j}$, at least one node in $I_{\overline{j}}$ must be removed from $\overline{j}$. □

4.2.5 Extended feasibility cuts (EFC)

In the previous section, we discussed MFCs in which the feasibility cuts are added for each hub. In this type of cut, we did not consider nodes which are not assigned to hub $\overline{j}$ in solution $\overline{x}$. However, in *extended feasibility cuts*, we restrict the nodes p whose effect on the capacity usage of the hub is higher than the maximum reduction of a node removal from this hub. In this case, even if we remove the node which gives the highest amount of reduction on the hub capacity usage, we cannot find a feasible solution for the hub $\overline{j}$ by adding this node p.

Lemma 4.2.3. *For a given integer solution $\overline{x}$ to* MP*, if constraint (4.4) is not satisfied for location j, then*

$$\sum_{i \in I_j^k} x_{ij} + \sum_{i \in \overline{I}_j^k} x_{ij} \leq \left|I_j^k\right| - 1 \;\; \forall j \in I_{inf}^k \tag{4.14}$$

is a valid feasibility cut for the master problem, where the set $\overline{I}_j^k$ is defined as follows:

$$\overline{I}_j^k = \left\{ p : \sum_{m \in I} (t_{mp} + t_{pm}) \geq \max_{i \in I : \overline{x}_{ij} = 1} \left(2 \cdot \sum_{m \in I} (t_{im} + t_{mi}) \right) \; and \;\; \overline{x}_{pj} = 0 \right\}$$

Proof. Assume that W items are assigned to hub location j. In the current solution, $\overline{x}_{1j} = \overline{x}_{2j} = ... = \overline{x}_{wj} = 1$. If total flow (adjacent traffic & backbone traffic) of these nodes (1,2..., w) are greater than the capacity of hub, M,

$$x_{1j} + x_{2j} + \ldots + x_{wj} \leq W - 1$$

is a valid feasibility cut.

In order to determine which nodes are included in the set $\overline{I}_j^k$, we compare two quantities as follows:

- *Maximum reduction:* When an assigned node (i) is removed from the hub j, maximum possible reduction on the usage of capacity of the hub due to the node i is

  $$2 \cdot \sum_{m \in I} (t_{im} + t_{mi}).$$

  To calculate the maximum possible reduction due to the removal of a node, we use the following idea.

  Consider the capacity constraints 4.4. In this constraint, the first term is incurred if a node i is assigned to the hub location j. The second term represents the total amount of backbone traffic between hub j and different hubs.

  - If a node i is removed from the hub $\overline{j}$, the first term in 4.4 is decreased by $\sum_{m \in I} (t_{im} + t_{mi})$.
  - For the second term in 4.4, we can use an upper bound on the amount of backbone traffic. If all the remaining nodes except i and $\overline{j}$ are assigned to different hubs, total amount of backbone flow related to $\overline{j}$ is $\sum_{m \in I} (t_{im} + t_{mi})$.

  By using this assumption, we can conclude that the maximum possible reduction is $2 \cdot \sum_{m \in I} (t_{im} + t_{mi})$ for node i.

- *Minimum increase:* In this case, we consider the node p which is not assigned to hub location j in the current master problem's solution (e.g., $\overline{x}_{pj} = 0$. The minimum increase on the usage of the capacity is $\sum_{m \in I} (t_{mp} + t_{pm})$. This amount of quantity is the first part of the nonlinear constraints 4.4 due to the $\overline{x}_{pj} = 1$ in the new solution.

When two quantities are compared (*maximum reduction & minimum increase*), if minimum increase due to the assignment of node p to hub location j is greater than equal to maximum reduction occurred due to the removal of node i, the node p is included in the set $\overline{I}_j^k$.

In this case, we try to show that we cannot assign node p to this hub besides the nodes that are currently assigned to hub j. This fact will be proved by contradiction.

Assume that

$$\sum_{i \in I_j^k} x_{ij} + \sum_{i \in \overline{I}_j^k} x_{ij} \leq \left|I_j^k\right| - 1 \;\; \forall j \in I_{inf}^k$$

is not a valid feasibility cut.

If this cut is not a valid cut, then there exists a cut in which W nodes are selected from $I_j^k \cup \{p\}$. W items which have the smallest weights are $(1, 2, ..., w)$, since $\sum_{m \in I}(t_{mp} + t_{pm}) \geq \max_{i \in I : \overline{x}_{ij}=1}\{2 \cdot \sum_{m \in I}(t_{im} + t_{mi})\}$.

Since $x_{1j} + x_{2j} + ... + x_{wj} \leq \left|I_j^k\right| - 1$ is a valid feasibility cut, we cannot find a cut such that $x_{1j} + x_{2j} + ... + x_{wj} + x_{pj} > \left|I_j^k\right| - 1$. This contradicts the assumption that we make in the beginning of the proof. □

4.3 Computational Results

In this study, we propose different variants of branch-and-check algorithm by making enhancement on the following parts:

- Initial valid inequalities added to the *MP*
  - VI(1): Inequality (4.8).
  - VI(2): Inequality (4.9).
- Feasibility cuts generated from the subproblem (*SP*)
  - Multiple Feasibility Cut: Inequality (4.11).
  - Extended Feasibility Cut: Inequality (4.14).
- The strategy for traffic bound calculations to find the backbone links

– Solving knapsack problems given in Algorithm 2 as IP or LP.

By taking into consideration all enhancement steps in the branch-and-check algorithm, we obtain eight variants of it. In order to see the effects of each improvement steps, we give the comparison results of these branch-and-check variants in this section. We use the notation ($BV_xT_yF_z$) to describe each decomposition alternative.

In $BV_xT_yF_z$ notation:

- V_x : represents the type of valid inequality that is added to the *MP* at the beginning. If $x = 1$, VI(1) is used, while $x = 2$, VI(2) is used. If we do not use any valid inequality in the master problem, then $x = 0$.

- T_y denotes the strategy of solving knapsack problems in traffic bound algorithm (2). If $y = LP$, linear programming relaxations of knapsack problems are used to find a lower bound on backbone traffic. On the other hand when $y = IP$, knapsack problems in Algorithm 2 are solved as IP.

- F_z : represents the type of feasibility cuts. When $z = MF$, *MFC* are added to the *MP*. If $z = EF$, *EFC* are used as feasibility cuts in the branch-and-check algorithm.

We compared different variants of Benders decomposition algorithm with MIQCP solver of IBM CPLEX. Different variants of Benders decompositions are also compared with each other to evaluate the effects of enhancement steps and find the best method for each instance. We conduct our experimental setting on a 64-bit machine @ 3.10 GHz and 16 GB of Ram. All methods are coded in IBM CPLEX 12.7.1 using *C++ Concert technology*. To implement our branch-and-check algorithm, we need to construct a single tree for the master problem and solve subproblem at each node of that tree. So, we use *Ilolazyconstraint callback* function that is available in IBM *CPLEX*. The summary results of the algorithms are given in this section.

4.3.1 Description of Instances

We use the well known AP data set from [1]. In the AP data set, each instance includes the number of nodes in the network, x and y coordinates of nodes, strictly positive flow between any pairs of nodes, capacity of hubs and their fixed costs. There are two levels for both capacity of hubs and fixed cost of them as loose (L) and tight (T). In our computational study, the cost multipliers for each unit flow are taken as $\alpha = 3$ and $\beta = 2$ for collection and distribution, respectively.

In the problem setting (QCCLP), demand of a node to itself is 0 (e.g., $t_{ii} = 0$). We use the same fixed cost values as AP data set. However, we use a single value for hub capacity which is given as M in the study. In order to see the effect of capacity on the performances of the solution methods, we determine two capacity levels such as 3000 and 4000 for tight (T) and loose (L) levels, respectively.

4.3.2 Comparison of MIQCP in CPLEX and Branch-and-Check Algorithms

We solved QCCLP formulation given in Section 4.1 by using MIQCP solver of CPLEX with default settings. Then, we evaluate the performances of our branch-and-check algorithm alternatives with MIQCP solver.

Table 4.1 gives the computational results for MIQCP solver and $BV_1T_{IP}F_{MF}$ on 20 nodes instances. Both methods solved all instances to optimum within given CPU time limit. Average CPU time required in $BV_1T_{IP}F_{MF}$ is less than the CPU time required by MIQCP. Moreover, average number of nodes opened in branch-and-bound tree of master problem is less than the branch-and-bound nodes in MIQCP. As we expected, the maximum number of feasibility cuts occurs in the setting where the hub capacities are small and fixed cost of opening them are high.

Table 4.2 gives the computational results of MIQCP solver and $BV_1T_{IP}F_{MF}$ for 25 nodes instances. $BV_1T_{IP}F_{MF}$ performs better than $MIQCP$ in terms of CPU time in 12 out of 20 instances. When we look at the average CPU times of MIQCP and $BV_1T_{IP}F_{MF}$, MIQCP is slightly better than $BV_1T_{IP}F_{MF}$. This result is mostly due to the setting where the hub capacities are 3000 and fixed cost is set to tight values.

Therefore, we focus on these instances.

Instance 6 in Table 4.2 is solved optimally in 618.63 sec. and 1,133.33 sec. by MIQCP and $BV_1T_{IP}F_{MF}$, respectively. The reason for a larger solution time with $BV_1T_{IP}F_{MF}$ when compared with MIQCP is increased number of feasibility cut. In order to reduce the number of feasibility cuts and computational time, we can use $BV_2T_{IP}F_{MF}$. When VI(2) is added to the MP, the required time for solving the model in $BV_2T_{IP}F_{MF}$ is 37.11 sec. As CPU time for calculating traffic bound is 97.17 sec., total CPU time for instance 6 is 135.6 sec. This implies that there is a significant effect of using VI(2) in the MP on overall CPU time. Both methods solve 19 instances to optimum. For instance 8 in Table 4.2 which is not optimally solved by MIQCP and $BV_1T_{IP}F_{MF}$, both methods finds the same solution when time limit is reached.

In 40 nodes instances, we select $BV_1T_{IP}F_{EF}$ and $BV_2T_{LP}F_{EF}$ to compare the performance of MIQCP for large and small size capacities, respectively. The computational results of MIQCP and branch-and-check alternatives are given in Table 4.3. We divide the analysis of computational study into two parts for 40 nodes instances.

When we look at the instances with *small capacities*, average CPU time in $BV_2T_{LP}F_{EF}$ is slightly better than the average CPU time in MIQCP. While MIQCP solves four instances to optimum, $BV_2T_{LP}F_{EF}$ solved eight instances optimally. Two instances (Instance 6 and Instance 8) in Table 4.3 are not solved optimally by any of the both methods $BV_2T_{LP}F_{EF}$ finds a better solution for instance 6. However, MIQCP finds a better solution for instance 8.

For the instances with *high capacities*, the average CPU time in $BV_1T_{IP}F_{EF}$ is slightly better than the average CPU time in MIQCP. The number of optimally solved instances for this treatment combination ,where the capacity values are 4000 and fixed costs are set to tight values, is the same in both methods. Therefore, we cannot conclude that one method outperforms the other. Any of both methods cannot solve the three instances (Instances 16, 17 and 18) optimally. While $BV_1T_{IP}F_{EF}$ finds a better solution than the solution found in MIQCP for instances 17 and 18, MIQCP finds a better solution for instance 16 in Table 4.3.

For 50 nodes instances, we select $BV_2T_{LP}F_{MF}$ variant to compare with MIQCP

and computational results are given in Table 4.4. For both high and low capacities, the performance of $BV_2T_{LP}F_{MF}$ is better than the performance of MIQCP in terms of average CPU time. For the instances with small capacity values, MIQCP and $BV_2T_{LP}F_{MF}$ solve seven instances optimally. When the results are compared in more detail, any method cannot find the optimal solution for instance 6 and 8 within given time limit. When compared the objective function values of the master problem at the time limit, $BV_2T_{LP}F_{MF}$ finds a better solution than MIQCP. While instance 7 cannot be solved optimally by MIOCP within given time limit, $BV_2T_{LP}F_{MF}$ finds the optimal solution for this instance. On the other hand, instance 9 is not solved by $BV_2T_{IP}F_{MF}$ while MIQCP solves this instance optimally.

One observation that can be concluded from 4.4, for small capacity values, the instances with tight fixed costs are more difficult than the instances with low fixed costs. As we expected, the average number of feasibility cuts that are added to the MP is higher in the settings with small capacity values and tight fixed cost values.

In order to show that we can solve large size instances to optimum by branch-and-check algorithm, we test our proposed algorithm on 100 nodes instances. The linearizations that are given in A and MIQCP solver cannot find an integer feasible solution within given time limit (2 hours) for 100 nodes instances. We think that the main reason behind this result is huge number of decision variables and constraints. Therefore, we only report the results of $BV_2T_{LP}F_{EF}$ in Table 4.5. Over 24 instances given in Table 4.5, 15 instances are solved to optimum by $BV_2T_{LP}F_{EF}$. Over these instances which could not be solved to optimum, maximum relative gap is 9% and average gap is 4%.

In our computational study, we observe that the instances with small capacity and high fixed costs are difficult instances. They require high number of feasibility cuts for the convergence. Therefore, the highest CPU time in the branch-and-check algorithm variant occurs in the treatment combination where the capacity is 3000 and fixed cost of concentrators are set to tight values.

When demand values of the terminals decrease, smaller number of concentrators (hubs) are opened in the optimal solution. As a result of this, the number of feasibility cuts decreases and overall CPU time of the branch-and-check algorithms also

decreases. When the demand values are low and the hub capacities are loose, we show that the instances cannot be solved optimally at the root node of the master problem in branch-and-check alternatives.

From our computational study, we observed that the instances with tight fixed costs are difficult to be solved optimally by any method (*MIQCP and Branch-and-check Algorithms*).

One of the most important results from our computational study is that when the number of nodes in the network increases, the performance of MIQCP becomes worse. The reason is that the number of decision variables in branch-and-check algorithms are less than the number of linearizations of the original model. As we generate the problem specific feasibility cuts and improved version of the master problem, branch-and-check can solve large size instances.

Table 4.1: Comparison results of MIQCP and $BV_1T_{IP}F_{MF}$ for 20 nodes

				MIQCP					$BV_1T_{IP}F_{MF}$						*Traffic Bound*		$BV_1T_{IP}F_{MF}$
Ins	**Nodes**	**Cap**	**FC**	**Obj**	**CPU**	**Nodes**	**Rgap**	**Hubs**	**Obj**	**CPU**	**Nodes**	**Rgap**	**Hubs**	**# of Fcuts**	$T_i(time)$	$T_{ij}(time)$	Total (CPU)
1	20	3000	L	243,550	10.2	1,528	0	3,9;11;14;19	243,550	1.4	751	0	3;9;11;14;19	451	0.47	47.22	1.85
2				211,700	14.6	2,776	0	3;11;13;14	211,700	2.1	2,124	0	3;11;13;14	694	0.63	52.53	2.71
3				201,947	35.5	15,309	0	3;9;11;14	201,947	5.1	4,483	0	3;9;11;14	1,341	2.28	63.64	7.36
4				184,475	5.9	487	0	6;11;14	184,475	1.9	1,340	0	6;11;14	466	0.56	69.23	2.44
5				173,226	8.7	1,588	0	3;11;14	173,226	3.7	3,021	0	3;11;14	1,062	0.86	66.61	4.53
6	20	3000	T	292,451	14.1	2,244	0	1;8;10;11;18	292,451	3.1	1,987	0	1;8;10;11;18	1,093	0.47	47.22	3.55
7				236,912	13.0	3,297	0	7;9;11;18	236,912	5.7	4,913	0	7;9;11;18	1,579	0.63	52.53	6.32
8				221,316	20.8	8,571	0	1;7;11;18	221,316	9.4	6,764	0	1;7;11;18	1,941	2.28	63.64	11.69
9				198,834	3.8	161	0	7;11;18	198,834	0.6	380	0	7;11;18	119	0.56	69.23	1.20
10				188,071	4.1	81	0	7;11;18	188,071	1.3	500	0	7;11;18	289	0.86	66.61	2.13
11	20	4000	L	223,990	3.9	132	0	3;9;11;14	223,990	0.6	571	0	3;9;11;14	172	0.64	68.05	1.27
12				193,755	5.1	326	0	6;11;14	193,755	0.7	575	0	6;11;14	216	0.97	65.64	1.67
13				181,317	4.7	900	0	3;11;14	181,317	0.4	135	0	3;11;14	112	0.50	60.13	0.91
14				162,588	0.1	0	0	7;14	162,588	0.0	0	0	7;14	1	0.27	64.31	0.29
15				155,237	0.1	0	0	7;14	155,237	0.0	0	0	7;14	0	0.48	56.25	0.50
16	20	4000	T	248,725	4.6	498	0	7;11;18	248,725	0.9	519	0	7;11;18	247	0.64	68.05	1.53
17				214,325	1.2	86	0	7;11;18	214,325	0.4	136	0	7;11;18	112	0.97	65.64	1.35
18				201,150	3.3	170	0	1;11;18	201,150	0.2	23	0	1;11;18	34	0.50	60.13	0.66
19				184,240	0.1	0	0	7;18	184,240	0.0	0	0	7;18	0	0.27	64.31	0.29
20				175,956	0.3	0	0	7;18	175,956	0.0	0	0	7;18	0	0.48	56.25	0.51
Average					8	1,908	0				1,411	0		496			3

Table 4.2: Comparison results of MIQCP and $BV_1T_{IP}F_{MF}$ for 25 nodes instances

				MIQCP					$BV_1T_{IP}F_{MF}$						*Traffic Bound*		$BV_1T_{IP}F_{MF}$
Ins	**Nodes**	**Cap**	**FC**	**Obj**	**CPU**	**Nodes**	**Rgap**	**Hubs**	**Obj**	**CPU**	**Nodes**	**Rgap**	**Hubs**	**# of Fcuts**	Ti (time)	Tij (time)	Total (CPU)
1	25	3000	L	242,304	22.32	5,000	0	3;11;14;18;24	242,304	3.58	3,111	0	3;11;14;18;24	725	1.38	97.17	4.96
2				204,369	17.02	2,095	0	3;14;16;24	204,369	2.97	1,808	0	3;14;16;24	671	1.42	141.23	4.39
3				195,595	13.50	1,628	0	3;14;16;24	195,595	4.59	4,657	0	3;14;16;24	594	0.88	122.47	5.47
4				180,858	11.50	1,278	0	9;11;24	180,858	5.20	4,954	0	9;11;24	1,058	0.7	113.14	5.90
5				130,111	0.17	0	0	8;24	130,011	0.02	0	0	8;24	0	0.86	99.4	0.88
6	25	3000	T	338,719	618.63	103,607	0	3;9;11;14;24	338,719	1,133.33	168,179	0	3;9;11;14;24	20,804	1.38	97.17	1,134.71
7				273,172	45.36	7,095	0	9;11;14;24	273,172	101.48	42,058	0	9;11;14;24	4,693	1.42	141.23	102.90
8				264,053	7,200.17	430,819	0.007	9;11;14;24	264,053	7,200.06	510,456	0.03	9;11;14;24	58,994	0.88	122.47	7,200.94
9				229,511	9.25	88	0	11;14;24	229,511	34.45	2,631	0	11;14;24	1,148	0.7	113.14	35.15
10				220,571	22.25	6,048	0	9;11;24	220,571	20.86	8,583	0	9;11;24	3,445	0.86	99.4	21.72
11	25	4000	L	222,479	5.05	144	0	3;9;16;18	222,479	0.64	136	0	3;9;16;18	81	0.28	102.81	0.92
12				191,488	4.34	36	0	9;11;18	191,488	0.64	274	0	9;11;18	106	0.64	111.41	1.28
13				181,654	7.44	472	0	9;11;18	181,654	0.81	249	0	9;11;18	158	0.8	120	1.61
14				168,549	1.69	17	0	8;18	168,549	0.19	1	0	8;18	9	0.44	122.78	0.63
15				130,011	0.19	0	0	8;24	130,011	0.03	0	0	8;24	0	0.16	6.55	0.19
16	25	4000	T	284,844	13.52	2,477	0	11,14;24	284,844	3.88	1,283	0	11;14;24	924	0.28	102.81	4.16
17				242,948	0.88	11	0	9;11;24	242,948	1.08	313	0	9;11;24	321	0.64	111.41	1.72
18				232,736	13.75	3,137	0	9;11;24	232,736	1.52	660	0	9;11;24	372	0.8	120	2.32
19				210,573	0.05	0	0	9;24	210,573	0.05	0	0	9;24	0	0.44	122.78	0.49
20				202,214	8.67	0	0	9;24	202,214	0.16	0	0	9;24	1	0.16	6.55	0.32
Average					401		0					0		4705			427

Table 4.3: Comparison of $MIQCP$ & $BV_2T_{LP}F_{EF}$ and $MIQCP$ & $BV_1T_{IP}F_{EF}$ for 40 nodes instances

					MIQCP					$BV_2T_{LP}F_{EF}$						*Traffic Bound*		$BV_2T_{LP}F_{EF}$
Ins	**Demand**	**Nodes**	**Cap**	**FC**	**Obj**	**CPU**	**Nodes**	**Rgap**	**Hubs**	**Obj**	**CPU**	**Nodes**	**Rgap**	**Hubs**	**# of Fcuts**	Ti (time)	Tij (time)	Total (CPU)
1	D	40			254,209	7,205.48	410,163	0.01	6;17;22;28;29	254,001	2,081.17	581,464	0	6;17;22;28;37	6,349	0.58	25.94	2,107.69
2	D/1.2	40	3000	L	214,462	7,200.30	2,279	0.02	14;15;29	214,462	1,554.94	123,538	0	6;22;25;28	309	0.45	29.66	1,585.05
3	D/1.3	40			203,248	7,057.31	459,181	0	6;22;25;28	203,248	451.97	73,667	0	6;22;25;28	7,310	0.41	20.28	472.66
4	D/1.4	40			187,990	446.41	51,133	0	10;22;28	187,990	29.13	6,654	0	10;22;28	1,312	0.41	30.94	60.48
5	D/1.5	40			175,597	67.34	2,836	0	14;25;29	175,597	8.27	832	0	14;25;29	261	0.41	28.06	36.74
6	D	40			353,868	7,205.83	271,761	0.08	6;14;19;22;35	345,786	7,200.02	2,865,600	0.03	6;14;19;22;35	1,565	0.58	25.94	7,226.54
7	D/1.2	40	3000	T	288,997	7,214.70	409,752	0.05	14;19;22;38	288,997	266.94	156,292	0	14;19;22;38	207	0.45	29.66	297.05
8	D/1.3	40			274,866	7,205.09	208,781	0.06	1;14;22;35	276,052	7,200.06	259,046	0.06	14;19;22;38	58,643	0.41	20.28	7,220.75
9	D/1.4	40			242,629	7,200.08	963,939	0.00	14;22;35	242,629	5.00	623	0	14;22;35	180	0.41	30.94	36.35
10	D/1.5	40			231,000	7,202.66	612,988	0	14;19;22	231,000	14.39	4,930	0	14;19;22	161	0.41	28.06	42.86
Average						**5,800.52**	**339,281.30**	**0.02**				**407,264.60**	**0.01**		**7,629.70**			**1,908.6**

					MIQCP					$BV_1T_{IP}F_{EF}$						*Traffic Bound*		$BV_1T_{IP}F_{EF}$
Ins	**Demand**	**Nodes**	**Cap**	**FC**	**Obj**	**CPU**	**Nodes**	**Rgap**	**Hubs**	**Obj**	**CPU**	**Nodes**	**Rgap**	**Hubs**	**# of Fcuts**	Ti (time)	Tij (time)	Total (CPU)
11	D	40			232,679	98.98	6,266	0	11;22;28;35	232,679	36.08	4,368	0	11;22;28;35	2,237	0.39	32.5	36.47
12	D/1.2	40	4000	L	198,607	44.98	2,238	0	11;22;38	198,607	59.27	13,619	0	11;22;28	2,635	0.56	29.66	59.83
13	D/1.3	40			185,104	51.13	2,892	0	11;22;28	185,104	28.34	4,303	0	11;22;28	1,908	0.38	20.28	28.72
14	D/1.4	40			171,534	14.22	186	0	14;28	171,534	1.70	26	0	14;28	86	0.38	28.95	2.08
15	D/1.5	40			161,281	1.11	0	0	14;28	161,281	0.09	0	0	14;28	0	0.42	29.27	0.51
16	D	40			307,782	7,205.90	266,568	0.03	1;14;19;38	317,359	7,200.05	364,837	0.07	14;22;35;38	47,768	0.39	32.5	7,200.44
17	D/1.2	40	4000	T	263,294	7,205.19	384,223	0.04	14;22;35	261,551	7,200.25	113,144	0.06	14;19;38	61,224	0.56	29.66	7,200.81
18	D/1.3	40			251,187	7,200.13	231,646	0.06	14;19;38	250,137	7,200.05	108,567	0.07	14;19;38	59,974	0.38	20.28	7,220.71
19	D/1.4	40			217,699	7.27	36	0	14;38	217,699	12.83	299	0	14;38	845	0.38	28.95	13.21
20	D/1.5	40			207,989	26.09	1,826	0	14;19	207,989	5.75	414	0	14;19	356	0.42	29.27	6.17
Average						**2,185.50**	**89,588.10**	**0.01**				**60,957.70**	**0.02**		**17,703.30**			**2,174.8**

Table 4.4: Comparison of $MIQCP$ & $BV_2T_{LP}F_{MF}$ for 50 nodes instances

					MIQCP					$BV_2T_{LP}F_{MF}$						*Traffic Bound*		$BV_2T_{LP}F_{MF}$
Ins	**Demand**	**Nodes**	**Cap**	**FC**	**Obj**	**CPU**	**Nodes**	**Rgap**	**Hubs**	**Obj**	**CPU**	**Nodes**	**Rgap**	**Hubs**	**# of Fcuts**	Ti (time)	Tij (time)	Total (CPU)
1	D	50			238,958	2,337.83	114,050	0	3;22;27;45;48	238,958	76.95	26,969	0	3;22;27;45;48	623	0.78	47.25	124.98
2	D/1.2	50	3000	L	205,671	415.72	22,492	0	3;27;33;48	205,671	6.92	303	0	3;27;33;48	103	0.52	45.92	53.36
3	D/1.3	50			193,248	636.89	39,631	0	3;22;27;48	193,248	7.22	525	0	3;22;27;48	121	0.63	25.34	33.19
4	D/1.4	50			180,876	323.45	11,356	0	17;22;48	180,876	9.45	337	0	17;22;48	167	0.39	42.92	52.76
5	D/1.5	50			168,983	70.81	997	0	17;22;48	168,983	4.31	124	0	17;22;48	54	0.25	36.39	40.95
6	D	50			369,636	7,207.86		0.06	3;22;27;45;48	367,627	7,200.03	646,000	0.0027	3;21;27;45;48	19960	0.78	47.25	7,248.06
7	D/1.2	50	3000	T	303,995	7,206.11	172,373	0.03	3;22;27;48	303,645	189.34	52,099	0	3;22;27;48	753	0.52	45.92	235.78
8	D/1.3	50			290,846	7,208.38		0.06	3;21;27;48	287,790	7,200.09	208,000	0.05	3;21;27;48	30336	0.63	25.34	7,226.06
9	D/1.4	50			260,197	2,420.75	49,957	0	17;21;48	260,965	7,200.06	311,287	0.012	21;27;48	37019	0.39	42.92	7,243.37
10	D/1.5	50			246,105	3,338.11	97,261	0	17;21;48	246,105	398.78	18,058	0	17;21;48	39903	0.25	36.39	435.42
Average						**3,116.59**	**63,514.63**	**0.02**			**2,229.32**	**126,370.20**	**0.01**		**8,953.50**			**2,269.39**

					MIQCP					$BV_2T_{LP}F_{MF}$						*Traffic Bound*		$BV_2T_{LP}F_{MF}$
Ins	**Demand**	**Nodes**	**Cap**	**FC**	**Obj**	**CPU**	**Nodes**	**Rgap**	**Hubs**	**Obj**	**CPU**	**Nodes**	**Rgap**	**Hubs**	**# of Fcuts**	Ti (time)	Tij (time)	Total (CPU)
11	D	50			223,098	70.25	1,165	0	3;27;32;48	223,098	5.77	148	0	3;27;32;48	97	0.77	45.17	51.71
12	D/1.2	50	4000	L	192,416	39.69	380	0	17;22;48	192,416	10.84	247	0	17;22;48	228	0.53	49.08	60.45
13	D/1.3	50			180,760	29.56	275	0	17;22;48	180,760	2.63	41	0	17;22;48	9	0.61	41.94	45.18
14	D/1.4	50			171,012	120.41	5,647	0	15;48	171,012	22.2	645	0	15;48	519	0.39	42.92	65.51
15	D/1.5	50			159,790	10.47	37	0	15;48	159,790	2.53	11	0	15;48	20	0.25	36.39	39.17
16	D	50			321,400	7,205.33	156,865	0.05	3,21;27;48	320,478	7200.03	384859	0.03	3;21;27;48	27463	0.77	45.17	7,245.97
17	D/1.2	50	4000	T	269,261	7,204.02	364,499	0.04	17;21;48	268,726	54.17	10103	0	3;27;48	984	0.53	49.08	103.78
18	D/1.3	50			256,362	7,204.92	179,987	0.06	17;21;48	256,362	7200.03	103505	0.02	17;21;48	55618	0.61	41.94	7,242.58
19	D/1.4	50			226,257	138.73	5,152	0	17;48	226,257	18.02	440	0	17;48	394	0.39	42.92	61.33
20	D/1.5	50			214,552	21.02	116	0	17;48	214,552	43.91	813	0	17;48	1119	0.25	36.39	80.55
Average						**2,204.44**	**71,412.30**	**0.02**			**1,456.01**	**50,081.20**	**0.025**		**8,645.10**			**1,499.62**

Table 4.5: Computational Results of $BV_2T_{LP}F_{EF}$ for 100 nodes

				$BV_2T_{LP}F_{EF}$						Traffic Bound		
Nodes	Demand	Cap	FC	Obj Value	CPU (sec)	Nodes	Rgap	FeasCut	Hubs	T(i)	T(ij)	Tot(CPU)
100	D	3000	L	256,413	7,200.10	155,546	0.01	13,601	5;34;64;76;91	1	92.92	7,294.02
	D/1.5			182,478	82.1	450	0	165	34;44;96	0.73	102.7	185.53
	D/2			144,014	7,200.50	7,504	0.0117	13,692	29;73	0.47	87.16	7,288.13
	D/2.5			123,846	11.2	0	0	0	29;73	0.34	95.45	106.99
	D/2.75			116,940	6.3	0	0	0	34;91	0.3	20.27	26.87
	D/3			110,151	3.2	0	0	0	73	0.7	90.36	94.26
100	D	3000	T	441,074	7,200.20	198,989	0.03	14,295	5;19;44;52;95	1	92.92	7,294.12
	D/1.5			296,490	7,200.60	54,494	0.06	19,634	5;52;86	0.73	102.7	7,304.03
	D/2			209,708	7,200.60	9,838	0.08	13,809	5;52	0.47	87.16	7,288.23
	D/2.5			156,071	8.5	9	0	1	5;52	0.34	95.45	104.29
	D/2.75			133,765	2.9	0	0	0	52	20.27	23.46	46.63
	D/3			64,331	3.2	0	0	0	52	0.7	90.36	94.26
100	D	4000	L	243,877	7,200.30	49,370	0.0012	16,780	5;35;64;96	0.98	94.25	7,295.53
	D/1.5			175,400	7,200.50	0	0.013	13,459	29;73	0.7	91.36	7,292.56
	D/2			142,361	2.8	0	0	0	29;73	0.31	20.11	23.22
	D/2.5			124,496	82.1	450	0	165	34;44;96	0.25	17.98	100.33
	D/2.75			110,693	2.9	0	0	0	73	0.28	18	21.18
	D/3			110,693	2.9	0	0	0	73	0.2	17.42	20.52
100	D	4000	T	399,844	7,200.60	70,053	0.09	18,753	5;39;52;95	0.98	94.25	7,295.83
	D/1.5			261,389	7,200.50	12,770	0.09	13,270	5;52	0.7	91.36	7,292.56
	D/2			172,067	2.9	0	0	0	52	0.31	20.11	23.32
	D/2.5			144,411	3	0	0	0	52	17.98	21.25	42.23
	D/2.75			125,973	2.9	0	0	0	52	0.28	18	21.18
	D/3			125,973	3.1	0	0	0	52	0.2	17.42	20.72

4.3.3 Analysis of Enhancement Steps on Branch-and-Check Algorithm

In this section, we analyze the effects of valid inequalities added to the master problem, the strategy of finding traffic bound values and different variants of feasibility cuts on computational time of the algorithm.

- *Effects of valid inequalities*

  In our computational study, we develop two problem specific valid inequalities and add one of them to the MP at the beginning. When we add valid inequalities to the MP, the feasible region of the master problem (MP) is reduced and it converges to the feasible region of the original nonlinear integer model (*QC-CLP*). We compare computational performance of BV_0F_{EC} and $BV_2T_{IP}F_{EC}$ to show the effects of using valid inequalities on the solution time of the algorithms. In BV_0F_{EC}, we do not add any valid inequalities to the MP. Therefore, in this branch-and-check variant, traffic bound algorithms are not used.

  As can be seen in Table 4.6, using VI(2) to the master problem reduces the average CPU time from 5,421.29 sec.to 2,329.35 sec. over ten instances. As

expected, average number of feasibility cuts is reduced significantly by using master problem with valid inequalities. In $BV_2T_{IP}F_{EC}$, average number of feasibility cuts is 14,640, while the average number of cuts in BV_0F_{EC} is 40,483.20.

Next, we compare the computational performances of using either VI(1) and using VI(2). The summary results of this comparison is given in Table 4.7. When the number of nodes increases in the network, total required time to calculate T_{ij} increases significantly. In Table 4.7, average CPU time gives the total time required in branch-and-check variants without adding the time for calculating lower bounds on the backbone traffic. $BV_2T_{IP}F_{MF}$ solves more instances to optimum for large size instances within given time limit for 25, 40 and 50 nodes. Moreover, the number of feasibility cuts for the convergence of the algorithm is less in $BV_2T_{IP}F_{MF}$. There is only one setting where $BV_1T_{IP}F_{MF}$ gives smaller CPU time. Because, computing T_{ij} values does not pay off. As a result of this, $BV_2T_{IP}F_{MF}$ outperforms $BV_1T_{IP}F_{MF}$ in terms of average CPU time.

Table 4.6: Impact of adding valid inequalities to the master problem (MP)

					$BV_2T_{IP}F_{EC}$						BV_0F_{EC}					
Ins	**Nodes**	**Demand**	**Cap**	**FC**	**Obj**	**CPU**	**Nodes**	**R.Gap**	**Hubs**	**F.Cuts**	**Obj**	**CPU**	**Nodes**	**R.Gap**	**Hubs**	**F.Cuts**
1	25	*D*	3000	T	338,719	1,133.33	168	0	3;9;11;14;24	20,804	338,719	3,672.39	263,690	0	3;9;11;14;24	37,838
2		*D/1.2*			273,172	101.48	42	0	9;11;14;24	4,693	273,172	7,200.13	172,857	0.013	9;11;14;24	74,438
3		*D/1.3*			264,053	7,200.06	510	0.03	9;11;14;24	58,994	264,629	7,200.03	214,452	0.029	3;11;14;24	58,575
4		*D/1.4*			229,511	34.45	2,631	0	11;14;24	1,148	229,511	117.09	3,303	0	11;14;24	1,820
5		*D/1.5*			220,571	0.63	84	0	9;11;24	5	220,571	24.84	8,255	0	9;11;24	3,914
6	40	*D*	3000	T	345,786	7,226.54	2,865,000	0.03	6;14;19;22;35	1,565	368,207	7,200.09	153,854	0.2	14;19;21;22;38	74,743
7		*D/1.2*			288,997	297.05	156,292	0	14;19;22;38	207	291,963	7,200.20	120,730	0.14	14;19;22;35	47,076
8		*D/1.3*			276,052	7,220.75	259,046	0.06	14;19;22;38	58,643	298,984	7,200.19	51,409	0.22	6;14;19;25	41,236
9		*D/1.4*			242,629	36.35	623	0	14;22;35	180	245,240	7,200.47	100,956	0.1	14;19;38	34,407
10		*D/1.5*			231,000	42.86	4,930	0	14;19;22	161	233,842	7,200.47	32,370	0.11	14;19;38	30,785
		Average				**2,329.35**	328,932.67	0.01		**14,640.00**		5,421.59	112,187.60	0.08		**40,483.20**

Table 4.7: Comparison Results of VI(1) and VI(2) for different number of nodes in the network

Nodes	Performance Measures	$BV_2T_{IP}F_{MF}$	$BV_1T_{IP}F_{MF}$
20	*# of instances solved to optimum*	20/20	20/20
	Average CPU time	2.88	**1.87**
	Average time for T_i	0.77	0.77
	Average time for T_{ij}	61.36	-
	Average Rel. Gap	0	0
	Average # of feasibility cuts	**149.4**	496.5
25	*# of instances solved to optimum*	20/20	19/20
	Average CPU time	**134.5**	426.9
	Average time for T_i	0.8	0.8
	Average time for T_{ij}	103.7	-
	Average Rel. Gap	0	0.03
	Average # of feasibility cuts	**531.15**	4,751.40
40	*# of instances solved to optimum*	16/20	8/20
	Average CPU time	**1,662.7**	4,626
	Average time for T_i	1.4	1.4
	Average time for T_{ij}	455.3	-
	Average Rel. Gap	0.04	0.07
	Average # of feasibility cuts	**7,970.1**	29,493.0
50	*# of instances solved to optimum*	15/20	7/20
	Average CPU time	**1,989.9**	4,803.7
	Average time for T_i	1.7	1.7
	Average time for T_{ij}	803.8	-
	Average Rel. Gap	0.03	0.1
	Average # of feasibility cuts	**6,701.3**	29,552.6

A detailed comparison of $MIQCP$, $BV_1T_{IP}F_{MF}$ and $BV_2T_{IP}F_{MF}$ are given in Tables A.2 and A.3.

Table 4.8 gives computational comparison of different branch-and-check alternatives and MIQCP for 40 and 50 nodes. In the last column of this table, we report the best method for each instance. Over 40 instances, $BV_2T_{LP}F_{MF}$ gives the smallest CPU time or finds the best solution within given two hours time limit. For 18 instances, we can conclude that $BV_2T_{LP}F_{MF}$ outperforms the other variants of branch-and-check algorithm and MIQCP. The reason is that in $BV_2T_{LP}F_{MF}$ has stronger valid inequalities (VI(2)) which are added to the master problem. As the number of nodes in the network is high, solving the knapsack problems in traffic bound algorithms as linear programs is advantageous in order to reduce required time for calculating backbone traffic. For

92.5 % of the instances given in Table 4.8, the branch-and-check variant is the best method, so it is concluded that solving QCCLP by a branch-and-check algorithm is advantageous.

Table 4.8: Computational comparison of different branch-and-check alternatives and MIQCP for 40 nodes and 50 nodes instances in terms of CPU time

Inst	Demand	Cap	FC	Nodes	$BV_1T_{IP}F_{MF}$	$BV_2T_{IP}F_{EF}$	$BV_2T_{LP}F_{EF}$	$BV_1T_{IP}F_{EF}$	$BV_1T_{LP}F_{EF}$	$BV_2T_{LP}F_{MF}$	MIQCP	Best Method
1	D	4000	L	40	49.9	366.9	46.4	37.2	41.2	44.3	99.0	$BV_1T_{IP}F_{EF}$
2	D/1.2			40	31.0	383.9	34.7	60.7	16.9	33.8	45.0	$BV_1T_{LP}F_{EF}$
3	D/1.3			40	25.7	487.6	30.8	29.5	26.1	44.5	51.1	$BV_1T_{IP}F_{MF}$
4	D/1.4			40	4.6	477.8	30.8	4.3	1.7	32.0	14.2	$BV_1T_{LP}F_{EF}$
5	D/1.5			40	1.1	625.1	29.8	1.1	0.5	29.8	1.1	$BV_1T_{LP}F_{EF}$
6	D	4000	T	40	7201.2	7556.0	7233.0	7201.6	7200.1	7232.9	7205.9	$BV_2T_{LP}F_{EF}$
7	D/1.2			40	7201.5	387.3	35.8	7201.3	7200.6	7230.3	7205.2	$BV_2T_{LP}F_{EF}$
8	D/1.3			40	7201.4	7671.6	7220.8	7201.1	7200.7	7220.8	7200.1	$BV_1T_{IP}F_{MF}$
9	D/1.4			40	143.6	476.1	29.7	13.5	12.2	166.1	7.3	**MIQCP**
10	D/1.5			40	62.2	630.8	38.1	7.6	5.5	95.1	26.1	$BV_1T_{LP}F_{EF}$
11	D	3000	T	40	7201.5	7563.7	7226.5	7201.3	7200.8	7226.6	7205.8	$BV_1T_{IP}F_{MF}$
12	D/1.2			40	7201.3	670.7	297.1	7201.3	7200.5	7230.2	7214.7	$BV_2T_{IP}F_{EF}$
13	D/1.3			40	7201.2	7622.6	7220.8	7201.2	7200.5	7220.8	7205.1	**MIQCP**
14	D/1.4			40	7201.3	461.5	36.4	7201.2	7200.5	7231.4	7200.1	$BV_2T_{LP}F_{EF}$
15	D/1.5			40	7202.0	631.6	42.9	7202.1	7200.5	7228.8	7202.7	$BV_2T_{LP}F_{EF}$
16	D	3000	L	40	7201.3	2574.7	2107.7	7202.0	7200.7	1946.2	7205.5	$BV_2T_{LP}F_{MF}$
17	D/1.2			40	7201.3	583.6	1585.1	7200.8	7200.7	164.3	7200.3	$BV_2T_{LP}F_{MF}$
18	D/1.3			40	7201.3	886.1	472.7	4765.0	7200.5	436.3	7057.3	$BV_2T_{LP}F_{MF}$
19	D/1.4			40	5798.2	480.1	60.5	2719.5	4300.5	56.2	446.4	$BV_2T_{LP}F_{MF}$
20	D/1.5			40	7202.0	632.2	36.7	7203.4	7200.4	33.7	67.3	$BV_2T_{LP}F_{MF}$
21	D	4000	L	50	371.9	710.1	51.6	434.5	723.3	6.5	70.3	$BV_2T_{LP}F_{MF}$
22	D/1.2			50	1276.3	796.0	61.6	629.5	248.7	11.4	39.7	$BV_2T_{LP}F_{MF}$
23	D/1.3			50	119.6	1000.8	44.3	158.2	65.7	3.2	29.6	$BV_2T_{LP}F_{MF}$
24	D/1.4			50	68.8	859.1	64.5	67.7	27.2	22.6	120.4	$BV_2T_{LP}F_{MF}$
25	D/1.5			50	4.7	914.8	38.4	4.7	1.8	2.8	10.5	$BV_2T_{LP}F_{MF}$
26	D	4000	T	50	7201.7	7991.6	7246.1	7201.8	7200.9	7200.8	7205.3	$BV_2T_{LP}F_{MF}$
27	D/1.2			50	7201.0	848.9	96.9	7201.1	7200.7	54.7	7204.0	$BV_2T_{LP}F_{MF}$
28	D/1.3			50	7201.9	8199.2	7242.9	7201.7	7200.8	7200.6	7204.9	$BV_1T_{IP}F_{EF}$
29	D/1.4			50	491.8	952.6	61.4	76.5	1227.7	18.4	138.7	$BV_2T_{LP}F_{EF}$
30	D/1.5			50	116.7	801.6	80.7	16.9	21.9	44.2	21.0	$BV_1T_{IP}F_{EF}$
31	D	3000	T	50	7202.1	7999.6	7248.1	7201.7	7200.9	7200.8	7207.9	$BV_1T_{IP}F_{EF}$
32	D/1.2			50	7202.8	1091.5	600.5	7201.6	7200.6	189.9	7206.1	$BV_2T_{LP}F_{MF}$
33	D/1.3			50	7201.8	8133.1	7226.0	7201.4	7200.7	7200.7	7208.4	$BV_1T_{IP}F_{EF}$
34	D/1.4			50	7201.7	8000.1	7243.4	7201.4	7200.5	7200.5	2420.8	**MIQCP**
35	D/1.5			50	7201.4	1180.9	374.3	7201.5	7200.3	399.0	3338.1	$BV_2T_{LP}F_{EF}$
36	D	3000	L	50	7202.0	730.5	152.1	7201.9	7200.8	77.7	2337.8	$BV_2T_{LP}F_{MF}$
37	D/1.2			50	7202.9	771.2	53.4	7203.2	7200.6	7.4	415.7	$BV_2T_{LP}F_{MF}$
38	D/1.3			50	7201.8	917.7	32.8	7201.8	7201.0	7.9	636.9	$BV_2T_{LP}F_{MF}$
39	D/1.4			50	7202.0	766.3	53.1	7201.9	7200.6	9.8	323.5	$BV_2T_{LP}F_{MF}$
40	D/1.5			50	7201.1	740.4	40.6	7201.1	7200.5	4.6	70.8	$BV_2T_{LP}F_{MF}$

- *Effects of the strategy of finding traffic bound*

  We will analyze the effects of strategy of finding traffic bound on overall performance of the branch-and-check algorithm into two parts such as either adding VI(1) to the MP or adding VI(2) to the MP.

  - *Adding VI(1) to the MP*

    In order to generate VI(1), we can solve the knapsack problems in Algorithm (2) by using two approaches such as using *LP* or *IP*. When we use EF in the algorithm, solving the knapsack problems as IP reduces average computational time calculated over 20 instances. In Table 4.9, average CPU time is decreased from 4540.6 to 4342.7 for 40 nodes. Similar result is also valid for 50 nodes instances. We cannot see any difference on the number of instances solved to optimality due to using different strategies for finding backbone traffic in the algorithm. One of the conclusions that can be obtained from the Table 4.9 is if we add VI(1) to master problem, solving the knapsack problems to find traffic bound as *LP* or *IP* does not have significant effect on computational time of the algorithms.

  - *Adding VI(2) to the MP*

    In the case where we add VI(2) to the MP, solving the linear programming relaxations of knapsack models in the traffic bound algorithm is advantageous. This result is expected, since total requires time to calculate traffic bound values for VI(2) is high for large size instances (e.g., 40 and 50 nodes instances). From Table 4.10, it is clear that $BV_2T_{LP}F_{EF}$ outperforms its counterpart $BV_2T_{IP}F_{EF}$ in terms of average computational time and the number of optimally solved instances. We give the detailed results of $BV_1T_{LP}F_{EF}$ for 40 nodes and 50 nodes instances in Tables A.4 and A.5, respectively. For more detailed results of $BV_2T_{IP}F_{EF}$, we refer the reader to Table A.6 for 40 nodes instances.

  When four different Benders variants ($BV_1T_{IP}F_{EF}$, $BV_1T_{LP}F_{EF}$, $BV_2T_{LP}F_{EF}$ and $BV_2T_{LP}F_{EF}$) are compared with each other, the best alternative is $BV_2T_{LP}F_{EF}$ in terms of average CPU time and number of optimally solved instances for both 40-nodes and 50-nodes. It is concluded that for the difficult instances with

Table 4.9: Comparison of using the strategy T_{IP} or T_{LP} in VI(1) for 40 and 50 nodes instances

Instances	***Performance Measures***	$BV_1T_{IP}F_{EF}$	$BV_1T_{LP}F_{EF}$
40 Nodes	*# of instances solved to optimum*	8/20	8/20
	Average CPU time (sec)	**4,327.7**	4,540.6
	Average no of feas cut	32,621.3	29,944.4
50 Nodes	*# of instances solved to optimum*	7/20	7/20
	Average CPU time (sec)	**4,750.5**	4,796.3
	Average no of feas cut	30,589.3	30,032.2

Table 4.10: Comparison of using the strategy T_{IP} or T_{LP} in VI(2) for 40 and 50 nodes instances

Instances	***Performance Measures***	$BV_2T_{IP}F_{EF}$	$BV_2T_{LP}F_{EF}$
40 Nodes	*# of instances solved to optimum*	16/20	16/20
	Average CPU time (sec)	2,058.5	**1,690.7**
	Average no of feas cut	8,802.0	9,078.1
50 Nodes	*# of instances solved to optimum*	15/20	15/20
	Average CPU time (sec)	2,667.6	**1,905.6**
	Average no of feas cut	8,656.3	5,959.3

high demand and low capacity values, adding strong valid inequalities to the MP increases the convergence speed of the algorithm.

4.3.3.1 Effects of feasibility cuts

In this section, we compare the computational performances of $BV_2T_{LP}F_{EC}$ and $BV_2T_{LP}F_{MF}$ in order to evaluate the effects of using different feasibility cuts. In the previous section, we show that $BV_2T_{LP}F_{EF}$ requires the smallest cpu time over four compared branch-and-check alternatives. Therefore, we test the performances of using either EF or MF on BV_2T_{LP} variant.

In Table 4.11, we give computational results of $BV_2T_{LP}F_{EF}$ and $BV_2T_{LP}F_{MF}$ for 40 and 50 nodes instances. We can solve 16 instances and 12 instances to optimum by using $BV_2T_{LP}F_{EF}$ and $BV_2T_{LP}F_{MF}$, respectively. Moreover, when two branch-and-check algorithms are compared in terms of average CPU time and the number of average feasibility cuts, using ***EF*** is better than using ***MF***. Therefore, we can conclude that the performance of $BV_2T_{LP}F_{EF}$ is better than the one of $BV_2T_{LP}F_{MF}$. This result implies that using extended feasibility

cuts has significant effect on the branch-and-check algorithms.

Table 4.11: Comparison Results of $BV_2T_{LP}F_{EF}$ and $BV_2T_{LP}F_{MF}$ in terms computational time and feasibility cuts for 40 nodes instances

Instances		$BV_2T_{LP}F_{EF}$		$BV_2T_{LP}F_{MF}$	
Cap	FCost	Total CPU (sec)	F.Cuts	Total CPU (sec)	F.Cuts
4000	L	46.44	753	**44.33**	684
		34.7	143	**33.75**	148
		30.83	408	44.52	1320
		30.83	62	32.02	66
		29.77	0	29.83	0
4000	T	7,231.17	49,957	7,231.09	51,342
		35.73	38	7,230.27	52,890
		7,220.9	53,515	7,220.86	42,745
		28.77	10	165.21	724
		35.13	378	92.12	299
3000	L	2,125.12	6,349	**1,963.67**	4,869
		1,583.58	309	**162.83**	325
		472.96	7,310	**436.6**	8,841
		64.68	1,312	**60.36**	862
		36.43	261	**33.35**	174
3000	T	7,226.54	1,565	7,226.57	75,969
		297.05	207	7,230.16	44,245
		7,220.75	58,643	7,220.77	42,219
		36.35	180	7,231.41	38,124
		42.86	161	7,228.78	39,119
Average		**1,691.5**	**9,078.05**	3,045.93	20,248.25

4.4 Comparison of Branch-and-check algorithm with Automatic Benders Implementation in IBM CPLEX 12.7.1

It is also possible to implement automatic Benders decomposition which is available in *IBM CPLEX 12.7.1* after the linearizations of *QCCLP*. We select *LSAHLP-1* and use automatic Benders decomposition (AutoBEND) tool in CPLEX. In Table *4.12*, we give the computational results of AutoBEND and one variant of branch-and-check algorithm for 20 instances. While only 6 instances are solved to optimality by using automatic Benders decomposition, we can solve 16 instances to optimality by a branch-and-check alternative. When no method cannot find the optimal solution within time limit, the bound obtained from the branch-and-check algorithm is better than the bound of *LSAHLP-1* with automatic Benders except the Ins 3 in Table 4.12. The rea-

son behind this result could be that we use feasibility cuts that are specific for the problem and adding strong valid inequalities to the *MP* improves the initial solution obtained from it. From these results, we can conclude that the performance of *AutoBEND* is not promising even for small instances.

Table 4.12: Comparison of Automatic Benders and branch-and-check algorithm for a subset of instances

					LSAHLP1 (Auto BEND)			$BV_1T_{IP}F_{EF}$		
Ins	Nodes	Demand	Cap	FC	Obj	CPU	R.Gap	Obj	CPU	R.Gap
1	25	D	3000	T	387,952	7,200.3	0.24	338,719	1,134.7	0
2		D/1.2			273,172	7,201.5	0.03	273,172	102.9	0
3		D/1.3			266,163	7,202.2	0.08	264,053	7,200.1	0.083
4		D/1.4			229,511	426.8	0	229,511	35.2	0
5		D/1.5			220,571	880.7	0	220,571	21.7	0
6	25	D	3000	L	246,807	7,200.9	0.07	242,304	4.96	0
7		D/1.2			204,369	1,314.1	0	204,369	4.39	0
8		D/1.3			195,595	4.468.9	0	195,595	5.37	0
9		D/1.4			180,858	653.6	0	180,858	5.90	0
10		D/1.5			130,011	122.5	0	130,011	0.02	0
					LSAHLP1 (Auto BEND)			$BV_2T_{IP}F_{MF}$		
Ins	Nodes	Demand	Cap	FC	Obj	CPU	R.Gap	Obj	CPU	R.Gap
1	40	D	3000	T	3,135,060	7,200.6	0.9	345,622	7,563.7	0.03
2		D/1.2			3,135,060	7,221.6	0.92	288,997	1,553.1	0
3		D/1.3			2,381,290	7,216.9	0.9	275,337	7,623.0	0.06
4		D/1.4			1,030,000	7,227.0	0.78	242,629	466.25	0
5		D/1.5			N.A	N.A	N.A	231,000	658.47	0
6	40	D	3000	L	538,027	7,217.1	0.58	254,001	2,461.9	0
7		D/1.2			223,345	7,262.7	0.12	214,462	538,94	0
8		D/1.3			803,13	7,217.0	0.77	203,248	1,225.08	0
9		D/1.4			914,55	7,218.4	0.81	187,99	541,6	0
10		D/1.5			197,311	7,216.3	0.16	175,597	647,6	0

4.5 Conclusion and Future Research

In this chapter, we study a quadratic capacitated concentrator location problem (*QCCLP*) where only capacity constraints of the hubs are nonlinear. The problem was proposed and developed a branch and cut algorithm for that problem by [53]. The novelty of our solution method comes from the fact that we handle nonlinearities within a decomposition based algorithm. By taking advantage of this decomposition, neither master problem nor the subproblem includes nonlinear terms. To the best of our knowledge, our study is the first where a Benders type decomposition algorithm (e.g., a branch-and-check algorithm) is applied for *QCCLP*. The computational study shows that adding strong valid inequalities to the master problem improves the solution obtained

from master problem. As generating strong valid inequalities may require huge amount of computational time for large size of instances, we should solve knapsack problems in traffic bound calculations as linear programs. After making enhancements on Benders decomposition, in only three instances, MIQCP's performance is better than the performance of branch-and-check alternatives over 40-node and 50-node instances. We can conclude that $BV_2T_{IP}F_{EF}$ and $BV_2T_{LP}F_{FC}$ are good alternatives in terms of several criteria such as number of optimally solved instances, CPU time, and relative gap. Moreover, we show that we can solve 100-node instances to optimality by using the decomposition algorithm, while MIQCP solver of CPLEX or linearizations result in out of memory error.

CHAPTER 5

QUADRATIC CAPACITATED HUB LOCATION PROBLEM (*QCHLP*)

In this chapter, we consider a quadratic capacitated hub location problem *(QCHLP)* where both objective function and constraints include nonlinear terms. The problem was first studied by [53]. QCHLP is an extension of QCCLP given in Chapter 4. The constraint sets are the same in both problems, QCCLP and QCHLP. In QCHLP, we are given a set of nodes and a traffic matrix in which the amount of traffic between any node pair is given. As in QCCLP, the decisions to be made are to select a subset of nodes as hubs and assign the remaining nodes to selected hubs in order to satisfy the demand of each origin-destination pair. The objective to minimize is the total cost consisting of three parts: *hub location cost*, *traffic routing cost between non-hub nodes and hub nodes* and *backbone traffic cost*.

Both problems (QCCLP, QCHLP) are in the class of single allocation hub location problems. When routing cost parameters are zero, QCHLP is reduced to QCCLP. As mentioned in Chapter 4, QCCLP is NP-hard. Since the simplified variant of the QCHLP is NP-hard, it is concluded that QCHLP is NP-hard, as well. Including nonlinear terms in the objective function of the model makes the problem more difficult to solve optimally within reasonable time limit.

In order to handle nonlinearities in the capacity constraints of QCCLP, we developed a branch-and-check algorithm in Chapter 4. For QCHLP, we propose a Benders decomposition type algorithm in order to cope with both nonlinear objective function and constraints. In this decomposition algorithm, the strategy to overcome nonlieairities in the constraints is the same as the strategy in QCCLP. To converge an optimal

solution, the algorithm must include the steps to handle nonlinear terms in the objective function.

The master problem of the decomposition algorithm in QCHLP is the same as the MP in the branch-and-check algorithm for QCCLP, except the MP of QCHLP includes an additional auxiliary variable in the objective function. This auxilary variable is used to converge the backbone traffic cost in the optimal solution.

The subproblems for both Benders decomposition algorithms for QCCLP and QCHLP include feasibility check for hub capacity constraints. If the solution obtained from master problem is found to be infeasible in the subproblem stage, (e.g., at least one hub's capacity is exceeded), feasibility cuts are added to the master problem.

In branch-and-check algorithm for QCCLP, the subproblem does not include evaluating the backbone traffic cost for a given master problem's solution. However, we need to evaluate the current backbone traffic cost in the subproblem of the Benders decomposition algorithm for QCHLP. When two Benders decomposition type algorithms (BDTAs) are evaluated in terms of the terminology in classical Benders decomposition techniques, BDTA for QCHLP includes both optimality and feasibility cuts. However, the branch-and-check algorithm for QCCLP includes only feasibility cuts.

Although Benders decomposition has been widely used to solve many different problems to optimum, its implementation on hub location problems is relatively rare. Especially for the problems in which hubs have capacity and single allocation is considered, Benders decomposition was not considered. We give the summary of the studies on the implementation of Benders decomposition for hub location problems in Table 5.1 to emphasize the our contribution to the literature.

Integer L-shaped algorithm was proposed by [54] for two stage stochastic programs in which first stage variables are binary and the subproblem could be a general problem. As in our decomposition method, all decision variables are binary, this method is applicable for *QCHLP*.

To the best of our knowledge, for QCHLP, the only exact solution algorithm which is a branch-and-cut algorithm was developed by [53].

Table 5.1: Benders decomposition implementations for hub location problems

Pr. Environment	Solution Strategy	Capacity Restrictions	Multiple Allocation	Single Allocation
Deterministic	*Iterative*	*Capacitated*	[66] [23]	
		Uncapacitated	[33] [22]	[16]
	Branch and Benders cut	*Capacitated*		***QCCLP*** ***QCHLP***
		Uncapacitated		
Stochastic	*Iterative*	*Capacitated*	[22]	[68]
		Uncapacitated		
	Branch and Benders cut	*Capacitated*	[59]	
		Uncapacitated	[58]	

In this chapter, our contributions can be summarized as follows:

- We develop an exact decomposition method for QCHLP. This decomposition approach is based on *Benders decomposition*.
- We propose problem specific feasibility and optimality cuts.
- In our decomposition algorithm, we don't use any linearizations for the nonlinear terms in the objective function and constraints.
- The performance of the optimality cuts which are specific for QCHLP is compared to that of MIQCP solver of IBM ILOG CPLEX.

In Section 5.1, we give problem definition and mathematical formulation for QCHLP. Then, we discuss our proposed decomposition algorithm which is based on Benders decomposition in Section 5.2. Next, we give the details of different optimality cuts used in the decomposition algorithm in Sections 5.2.3 and 5.2.3.1. Then, we give the computational results in Section 5.3.

5.1 Problem Definition and Mathematical Formulation

We are given a set of nodes I. Let t_{im} be the demand or flow sent from node i to node m. There is a corresponding fixed cost f_i incurred if node i is selected as a hub

location. d_{ij} represents the distance between nodes i and j. In *QCHLP*, each hub has the same capacity, M. In the objective function, α and β represent collection and distribution unit parameters, respectively as in QCCLP. In addition to the parameters given for QCCLP, there is an additional parameter R_{jl} which is a routing cost of one unit backbone flow transferred between different hubs j and l in QCHLP.

QCHLP can be defined as follows. A subset of node set nodes I to be chosen as hub locations and the remaining nodes are assigned to these hub locations. The objective to minimize is the sum of *fixed cost of opening hubs*, *routing cost between terminal nodes & hub nodes* and *backbone traffic cost*. Each node must be assigned to exactly one hub location.

x_{ij} represents the 0-1 decision to assign node i to node j. If node j is selected as a hub, then $x_{jj} = 1$, otherwise $x_{jj} = 0$. By using these parameters and the decision variables, we give the mathematical model [53] below:

(QCHLP)

$$\min \sum_{j\in I} f_j x_{jj} + \sum_{i\in I}\sum_{j\in I:i\neq j} d_{ij}x_{ij}(\sum_{m\in I}(\alpha t_{im} + \beta t_{mi}))+ \sum_{i\in I}\sum_{j\in I}\sum_{m\in I}\sum_{l\in I:j\neq l} R_{jl}t_{im}x_{ij}x_{ml} \tag{5.1}$$

$s.t.$

$$\sum_{j\in J} x_{ij} = 1, \qquad \forall i \in I. \tag{5.2}$$

$$x_{ij} \leq x_{jj}, \qquad \forall i,\ j \in I, i \neq j. \tag{5.3}$$

$$\sum_{i\in I}\sum_{m\in I}(t_{im} + t_{mi})x_{ij} + \sum_{i\in I}\sum_{m\in I}(t_{im} + t_{mi})x_{ij}(1 - x_{mj}) \leq Mx_{jj}, \forall j \in I. \tag{5.4}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, \quad j \in I. \tag{5.5}$$

In this formulation, the objective function includes three terms: fixed cost of selecting the nodes as hubs, routing cost between non-hub nodes and hub nodes and the cost of backbone traffic. Constraints (5.2) guarantee that each node is assigned to exactly one hub. Constraints (5.3) ensure that node i can be assigned to node j only if j is a hub. Constraints (5.4) ensure that total flow that uses the capacity of hub

j cannot exceed the capacity of hub which is defined as M. In Constraints (5.4), $\sum_{i\in I}\sum_{m\in I}(t_{im}+t_{mi})$ is the total flow of nodes that are assigned to hub j and $\sum_{i\in I}\sum_{m\in I}(t_{im}+t_{mi})x_{ij}(1-x_{mj})$ is the flow occurred on backbone links. Constraints (5.5) are the sign restrictions on decision variables.

Regardless of the objective function, if the model includes quadratic constraints, this resulting model is a mixed integer quadratically constrainted programming *(MIQCP)*.

We will give the Benders decomposition type algorithm for this nonlinear integer formulation of QCCLP in the Section 5.2.

5.2 A Benders Decomposition Type Algorithm (BDTA) for the QCHLP

We develop a Benders decomposition method to cope with both nonlinear objective function and constraints without using any linearizations. Nonlinearities in this formulation (QCHLP) are due to the multiplications of binary variables. More specifically, in the objective function of the (QCHLP), there are nonlinear terms: $x_{ij}\cdot x_{ml}$. Similarly, in the capacity constraints, the amount of backbone traffic between hub j and different hub l is calculated by the multiplications of binary variables as $(x_{ij}\cdot(1-x_{mj}))$.

The idea behind the decomposition algorithm can be summarized as follows:

We remove the nonlinear capacity constraints and nonlinear objective function component from QCHLP. To replace the nonlinear term in the objective function, we add auxilary variable, θ. Similarly, we replace the nonlinear terms in the capacity constraints with a lower bound on the amount of backbone traffic. Therefore, the resulting model, namely the MP, is a relaxation of QCHLP. After finding the values of decision variables ($\overline{x}_{ij}$, $\overline{x}_{jj}$), we can evaluate two following items:

- *Evaluating the current backbone traffic cost as:*

$$\sum_{i\in I}\sum_{m\in I}\sum_{j\in I}\sum_{l\in I:j\neq l}R_{jl}t_{im}\overline{x}_{ij}\overline{x}_{ml}$$

- *Evaluating the amount of flow that uses the hub capacity as:*

$$\sum_{i\in I}\sum_{m\in I}(t_{im}+t_{mi})\overline{x}_{ij}+\sum_{i\in I}\sum_{m\in I}(t_{im}+t_{mi})\overline{x}_{ij}(1-\overline{x}_{mj})\leq M\overline{x}_{jj}.$$

The feasibility cut generation step is the same as branch-and-check algorithm for QCCLP. In addition to this step, we generate optimality cuts and add them to the MP by comparing the value

$$\sum_{i\in I}\sum_{m\in I}\sum_{j\in I}\sum_{l\in I:j\neq l}R_{jl}t_{im}\overline{x}_{ij}\overline{x}_{ml}$$

and auxilary variable, θ.

To sum up, we develop a Benders decomposition type algorithm in which both feasibility and optimality cuts are generated for QCHLP. The differences and similarities between decomposition algorithms proposed for QCCLP in Chapter 4 and QCHLP in this chapter are summarized in Table 5.2.

Table 5.2: Comparing the Decomposition Algorithms for QCCLP and QCHLP

Problem[1]	**Method** [2]	**Feas Cut (When?)** [3]	**Feas Cut (How?)** [4]	**Opt Cut (When?)**[5]	**Opt Cut (How?)**[6]
QCCLP	*Branch-and-check algorithm*	$\exists\ j$: Constraint (5.4) is not satisfied for $\overline{x}$	$\sum_{i \in I_j^k} x_{ij} \leq \left\|I_j^k\right\| - 1, \forall j \in I_{inf}^k \quad \forall k = 1, 2..., k-1.$	No	No
QCHLP	*Benders Decomposition Type Algorithm*	$\exists\ j$: Constraint (5.4) is not satisfied for $\overline{x}$	$\sum_{i \in I_j^k} x_{ij} \leq \left\|I_j^k\right\| - 1, \forall j \in I_{inf}^k \quad \forall k = 1, 2..., k-1.$	$\theta < \sum_{i \in I} \sum_{m \in I} \sum_{j \in I} \sum_{l \in I} R_{jl} t_{im} \overline{x}_{ij} \overline{x}_{ml}$	Traffic Bound Based Cuts or LL optimality Cuts

(1): *Gives the problem type*

(2): *Defines the decomposition method for QCCLP and QCHLP.*

(3): *Defines the time when feasibility cuts are added to the master problem.*

(4): *Gives the type of the feasibility cuts used in the decomposition algorithm. Note that, MFC are given as examples, EFC could be used to replace MFC. In the computational study, we used EFC.*

(5): *Defines the time where optimality cuts are added to the master problem. Since the subproblem phase in the decomposition algorithm for QCCLP doesn't include backbone traffic cost, no optimality cuts are added to the master problem.*

(6): *Defines the time when optimality cuts are added to the master problem. In order to close the gap between the values of θ and current backbone traffic cost, we add optimality cuts in the decomposition algorithm for QCHLP.*

5.2.1 Master Problem

When the nonlinear constraints and the objective function are removed from QCHLP, we end up with a relax version of the QCHLP. This problem is the MP of BDTA.

The master problem that we consider in this decomposition algorithm is as follows:

$$\min \sum_{j\in I} f_j x_{jj} + \sum_{i\in I}\sum_{j\in I:i\neq j} d_{ij}x_{ij}(\sum_{m\in I}(\alpha t_{im} + \beta t_{mi})) + \theta$$

(MP) s.t.

$$(5.2), (5.3), (5.5)$$

$$\theta \geq \eta_{l_0} - \sum_{i\in I} \eta_{l_i} x_i \quad \forall l \in \omega_{opt}. \tag{5.6}$$

$$\sum_{i\in I} \mu_{l_i} x_i \geq \mu_{l_0} \qquad \forall l \in \omega_{feas}. \tag{5.7}$$

$$\theta \geq 0. \tag{5.8}$$

In this formulation, ω_{opt} is the set of optimality cuts generated. Similarly, ω_{feas} is the set of feasibility cuts. η- and μ- values are coefficients used in the cuts. In the first step of the algorithm, *MP* is solved without adding any feasibility and optimality cut.

The decisions in the MP are to determine the nodes which are selected as hubs and the assignments of the terminal nodes (non-hub nodes) to hub nodes. However, VI(1) and VI(2) which are given for QCCLP in Chapter 4 are also valid for QCHLP. Traffic Bound algorithms to obtain lower bounds on backbone links are also the same as QCCLP.

5.2.2 Subproblem and Solution Approach

For a given integer master problem's solution vector ($\overline{x}$), we check the feasibility of the following inequality for each opened hub:

$$\sum_{m\in I}\sum_{i\in I}(t_{im} + t_{mi})\overline{x}_{ij} + \sum_{i\in I}\sum_{m\in I}(t_{im} + t_{mi})\overline{x}_{ij}(1 - \overline{x}_{mj}) \leq M\overline{x}_{jj} \;\; \forall j \in I : \overline{x}_{jj} = 1. \tag{5.9}$$

After checking the inequality, two cases exist that we encounter:

- *Case 1: $\overline{x}$ is an infeasible solution in terms of capacity constraints:*

  If inequality (5.9) is not satisfied for an open hub j in solution $\overline{x}$, we need to add *feasibility cuts* to cut off this current solution. In this case, we have two alternative feasibility cuts. These cuts are the same as the cuts given in Chapter 4 for QCCLP.

  MFC:

$$\sum_{i \in I_j^k} x_{ij} \leq \left|I_j^k\right| - 1, \forall j \in I_{inf}^k \quad \forall k = 1, 2..., k-1. \tag{5.10}$$

  EFC:

$$\sum_{i \in I_j^k} x_{ij} + \sum_{i \in \overline{I}_j^k} x_{ij} \leq \left|I_j^k\right| - 1 \;\; \forall j \in I_{inf}^k \;\; \forall k = 1, 2..., k-1. \tag{5.11}$$

- *Case 2: $\overline{x}$ is a feasible solution:*

  If inequality (5.9) is satisfied for each hub, we evaluate backbone traffic cost given in Equation (5.12) by using the values of solution vector $\overline{x}$:

$$\theta_r(\overline{x}) = \sum_{i \in I} \sum_{j \in I} \sum_{m \in I} \sum_{l \in I: j \neq l} R_{jl} t_{im} \overline{x}_{ij} \overline{x}_{ml}. \tag{5.12}$$

  If $\theta < \theta_r(\overline{x})$, then we add optimality cuts to the MP. Optimality cuts are used to converge the exact value of the backbone traffic cost. In this decomposition algorithm for QCHLP, we propose an optimality cut which is problem specific and uses the lower bound values on backbone links. We call this specific optimality cut *traffic bound based cuts*. We give the details of specific optimality cuts in Section 5.2.3.

5.2.3 *Traffic Bound Based Optimality Cuts* (TBB Optimality Cuts)

When we find an integer feasible solution ($\overline{x}$) which satisfies the capacity constraints for each hub, we can calculate the value of current backbone traffic cost. This value is denoted by $\theta_r(\overline{x})$ in the optimality cut 5.13. When a value in the solution vector $\overline{x}$ changes (e.g., from zero to one or one to zero), the amount of reduction and increase on the objective function value can be calculated. This special optimality cut that

we propose for QCHLP is based on this idea. The general structure of the special optimality cut is given as:

$$\theta \geq \theta_r(\overline{x}) - \sum_{i \in I} \sum_{j \in I: \overline{x}_{ij}=1} A_{ij}(1 - x_{ij}) + \sum_{i \in I} \sum_{j \in I: \overline{x}_{ij}=0} B_{ij} x_{ij} \tag{5.13}$$

In Inequality (5.13), A_{ij} and B_{ij} are non-negative coefficients. The second term in 5.13 gives the *maximum reduction* on the current backbone traffic cost ($\theta_r(\overline{x})$). Last term represents the *minimum amount of increase* on the current backbone traffic cost ($\theta_r(\overline{x})$). If one of the values of $\overline{x}_{ij}$ vector changes from one to zero, the cost coefficient for this change is A_{ij} in the optimality cut. This coefficient defines the one-unit reduction on the value of current backbone traffic cost ($\theta_r(\overline{x})$) by changing the value of $\overline{x}_{ij}$ from one to zero. On the contrary, if a node i is not assigned to a hub j in the current solution and this node will be assigned to this hub after changing the solution, the cost coefficient for the amount of increase on the objective function value is B_{ij}. To estimate A_{ij} and B_{ij} values in the optimality cut, we focus on the *maximum amount of decrease* and the *minimum amount of increase* on ($\theta_r(\overline{x})$). We will give the detail derivations of these coefficients in Proposition 5.2.1 and Proposition 5.2.3. Calculations of A_{ij} are given in Proposition 5.2.1, while B_{ij} calculations are given in Proposition 5.2.3.

Proposition 5.2.1. *Suppose that $\overline{x}_{ij} = 1$ in a solution $\overline{x}$ to QCHLP. Assume that the solution is perturbed to value $\overline{x}_{ij} = 0$. Then the maximum possible reduction on backbone traffic cost ($\theta_r(\overline{x})$) due to the change in the value of x_{ij} is:*

$$\sum_{i \in I} \sum_{j \in I: \overline{x}_{ij}=1} \sum_{m \in I} \max_{l \in I} \{R_{jl}\}(t_{im} + t_{mi})(1 - x_{ij})$$

Proof. Consider the nodes that are assigned to the hub j (e.g., $\overline{x}_{ij} = 1$) in the given MP's solution. Assume that we remove this node from the hub by adding optimality cuts.

If this node is not assigned to hub j in the new solution, maximum total traffic that pass through the hub locations j and the other hubs is $\sum_{m \in I}(t_{im} + t_{mi})$. In this case, we assume that the remaining nodes except the node i are assigned to different hubs rather than hub j. Therefore, all traffic from node i to other nodes and from all nodes to this node i will use the backbone links. As we try to find the maximum reduction

due to the range of $\overline{x}$ values from 1 to 0 on $\theta(\overline{x})$, we multiply the total flow with the maximum routing cost between j and the remaining hub nodes $\max_{l\in L}\{R_{jl}\}$ If the current solution $\overline{x}$ is not changed, there will be no cost reduction on the value of $\theta(\overline{x})$. Therefore, we obtain the following part for the maximum reduction on the objective function value of the subproblem.

$$\sum_{i\in I}\sum_{j\in I:\overline{x}_{ij}=1}\sum_{m\in I}\max_{l\in I}\{R_{jl}\}(t_{im}+t_{mi})(1-x_{ij}).$$

□

Example 5.2.2. *Consider a network with 9 nodes. Nodes are given as 1,2,3,4,5 and j, l_1, l_2 and l_3. In this example, j, l_1, l_2, l_3 are hubs and the nodes that are assigned to these hubs are given with the following sets:*

$I_j = \{1, j\}$, $I_{l_1} = \{2, 3, l_1\}$, $I_{l_2} = \{l_2, 4\}$ *and* $I_{l_3} = \{l_3, 5\}$.

To write the backbone traffic, we need to consider the second part of the Inequality (5.9).

The amount of flow on backbone traffic occurred between hub j and the remaining hubs l_1, l_2 and l_3, A_j is:

$$\begin{aligned}A_j =&(t_{l_11}+t_{1l_1})+(t_{12}+t_{21})+(t_{13}+t_{31})+(t_{1l_2}+t_{l_21})+(t_{14}+t_{41})+\\&(t_{1l_3}+t_{l_31})+(t_{15}+t_{51})\end{aligned}$$

When we multiply this quantity with the maximum routing cost which is the maximum of $\max\{R_{jl_1}, R_{jl_2}, R_{jl_3}\} = R_j(max)$, we obtain $A_jR_j(max)$ as the amount of cost reduction occurring on $\theta_r(\overline{x})$.

Proposition 5.2.3. *Suppose that $\overline{x}_{ij} = 0$ in a given MP solution ($\overline{x}$). When this solution is perturbed to value $\overline{x}_{ij} = 1$, the minimum increase on the backbone traffic traffic cost $\theta_r(\overline{x})$ due to the change in the value of x_{ij} is:*

$$\sum_{i\in I}\sum_{j\in I:\overline{x}_{ij}=0}\min_{l\in I}\{R_{jl}\}T_{ij}x_{ij}.$$

Proof. In this part of the proof, we will consider the nodes which are not assigned to hub j ($\overline{x}_{ij} = 0$). When the optimality cuts are added to the MP, assume that the value

of $\overline{x}_{ij}$ changes zero to one. This implies, the node (i) is assigned to the hub node (j) in the new solution different than $\overline{x}$. We need to estimate the minimum amount of traffic increase on the backbone links due to this change in the solution vector $\overline{x}$.

To find a lower bound on the backbone links, we can use T_{ij} values. Remember the discussion about obtaining T_{ij} given in Traffic Bound Algorithm in Chapter 4. T_{ij} values are the lower bound values between hub j and the remaning hubs, when node i is assigned to hub j. If we multiply this amount of increase on the backbone traffic with minimum routing cost value, minimum amount of increase on $\theta_r(\overline{x})$ is calculated as

$$\sum_{i \in I} \sum_{j \in I: \overline{x}_{ij}=0} \min_{l \in I}\{R_{jl}\} T_{ij} x_{ij}.$$

□

By using Propositions (5.2.1) and (5.2.3), we find α_{ij} and β_{ij} values as:

$$A_{ij} = \sum_{m \in I} \max_{l \in I}\{R_{jl}\}(t_{im} + t_{mi}).$$

$$B_{ij} = \min_{l \in I}\{R_{jl}\} T_{ij}.$$

When plugging α_{ij} and β_{ij} values in Inequality 5.13, we obtain the optimality cut as follows:

$$\theta \geq \theta_r(\overline{x}) - \sum_{i,m \in I} \sum_{j \in I: \overline{x}_{ij}=1} \max_{l \in I}\{R_{jl}\}(t_{im} + t_{mi})(1 - x_{ij}) + \sum_{i \in I} \sum_{j \in I: \overline{x}_{ij}=0} \min_{l \in I}\{R_{jl}\} T_{ij} x_{ij} \tag{5.14}$$

Such kind of cuts that are specific for the problem can also be seen in some studies ([64] and [77]) in the literature. In the study ([64]), they consider a stochastic facility location problem in which uncertainty comes from discrete scenarios. As classical Benders decomposition implementation is not suitable for the problem due to the fact that the second stage problem includes integer variables, they propose an integer decomposition. In this decomposition algorithm, they develop special optimality cuts for the problem and compare its performance with the general cuts proposed by ([54]).

5.2.3.1 *Laporte & Louveaux Inequalities(LL cuts)*

In the literature, a decomposition method which is called Integer L-shaped algorithm was proposed by [54]. As mentioned before, this algorithm is firstly developed to solve two stage stochastic integer problems. They develop general purpose optimality cuts that are valid for any problem with a pure binary master problem. Due to the generality of this cut, the performance of the cut could generally be weak in the implementations. To apply an integer L-shaped algorithm for a problem, the subproblem should be solved for a given master problem's solution. After solving the subproblem, obtained objective function value is used in optimality cut. The general optimality cut which is also known as Laporte and Louveaux's cut (LL cut) is given as:

$$\theta \geq (\theta_r(\overline{x}) - L) \cdot \{ \sum_{i \in S(\overline{x})} x_i - \sum_{i \notin S(\overline{x})} x_i - |S| + 1\} + L. \tag{5.15}$$

In this optimality cut given in Inequality (5.15), $S(\overline{x}) = \{(i,j) : \overline{x}_{ij} = 1\}$ and the complement of this set $\overline{S}(\overline{x}) = \{(i,j) : \overline{x}_{ij} = 0\}$. L is a global lower bound for the objective function value of the subproblem.

There can be two cases after adding the optimality cut 5.15 to the MP. This cut either changes the current solution or enforce θ value to take the subproblem's objective function value (θ_r). When the solution is enforced to change by adding this cut, θ value will be equal to L. Therefore, the performance of the inequality 5.15 depends on the L value. This optimality cut is also used for several problems (e.g., [60], [64] and [56]).

Finding a lower bound (L) on the cost of backbone traffic

In order to find a lower bound (L) on the cost of backbone traffic, we consider a subset of constraints in the original model (QCHLP) and we don't use any linearizations for the nonlinear capacity constraints. The capacity relation between amount of flow and hubs is considered in traffic bound algorithms. These algorithms whose details are given in Chapter 4 are used to obtain lower bound on the amount of flow on backbone links. T_j and T_{ij} values obtained from traffic bound algorithms are used as parameters in the following mathematical formulation (LBP).

$$\min \sum_{i \in I} \sum_{j \in I: j \neq i} T_{ij} x_{ij} R_j^{min} + \sum_{j \in I} T_j x_{jj} R_j^{min}$$

(LBP) s.t.

$$\sum_{i \in I} x_{ij} = 1, \quad \forall j \in I. \tag{5.16}$$

$$x_{ij} \leq x_{jj}, \qquad \forall i, j \in I. \tag{5.17}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, j \in I. \tag{5.18}$$

The objective function in LBP is to minimize total backbone traffic routing cost in the network. Since we assume that all backbone traffic is routed on the backbone link with a smallest routing cost $R_j^{min} = min_{l \in L}\{R_{jl}\}$, we multiply the minimum amount of backbone traffic on backbone links with R_j^{min}. The second term in the objective function is incurred if a node j is a hub. Constraints (5.2) are single assignment constraints. Constraints (5.17) gives a logic relation between the assignment decisions and hub location decisions. Lastly, Constraints (5.18) are binary sign restrictions.

It is easy to see that the model has totally unimodular property. Therefore, we can relax the integrality restrictions of the decision variables. As can be concluded from the calculations of lower bounds in Traffic Bound algorithms, $T_{ij} \geq T_j$ for each i and j pair. Since the model includes single assignment constraints, the model is enforced to select each node as a hub node to minimize the objective function.

Algorithm 3 A Benders Decomposition Type Algorithm (BDTA) for QCHLP

Solve the *MP* and find an integer feasible solution ($\overline{x}$)

For each opened hub location j in solution $\overline{x}$:

if $\sum_{i\in I}\sum_{m\in I}(t_{im}+t_{mi})\overline{x}_{ij}+\sum_{i\in I}\sum_{m\in I}(t_{im}+t_{mi})\overline{x}_{ij}(1-\overline{x}_{mj})\leq M\overline{x}_{jj}$ **then**

    if $\theta < \theta_r(\overline{x})$ **then**

        Add one of the optimality cuts given in (5.14) and (5.15).

    end if

    if The current solution is the best solution so far for the QCHLP **then**

        Update the incumbent solution.

    end if

else

    Add one of the feasibility cuts given in Inequality (4.11) and Inequality (4.14) to the *MP* and continue solving MP.

end if

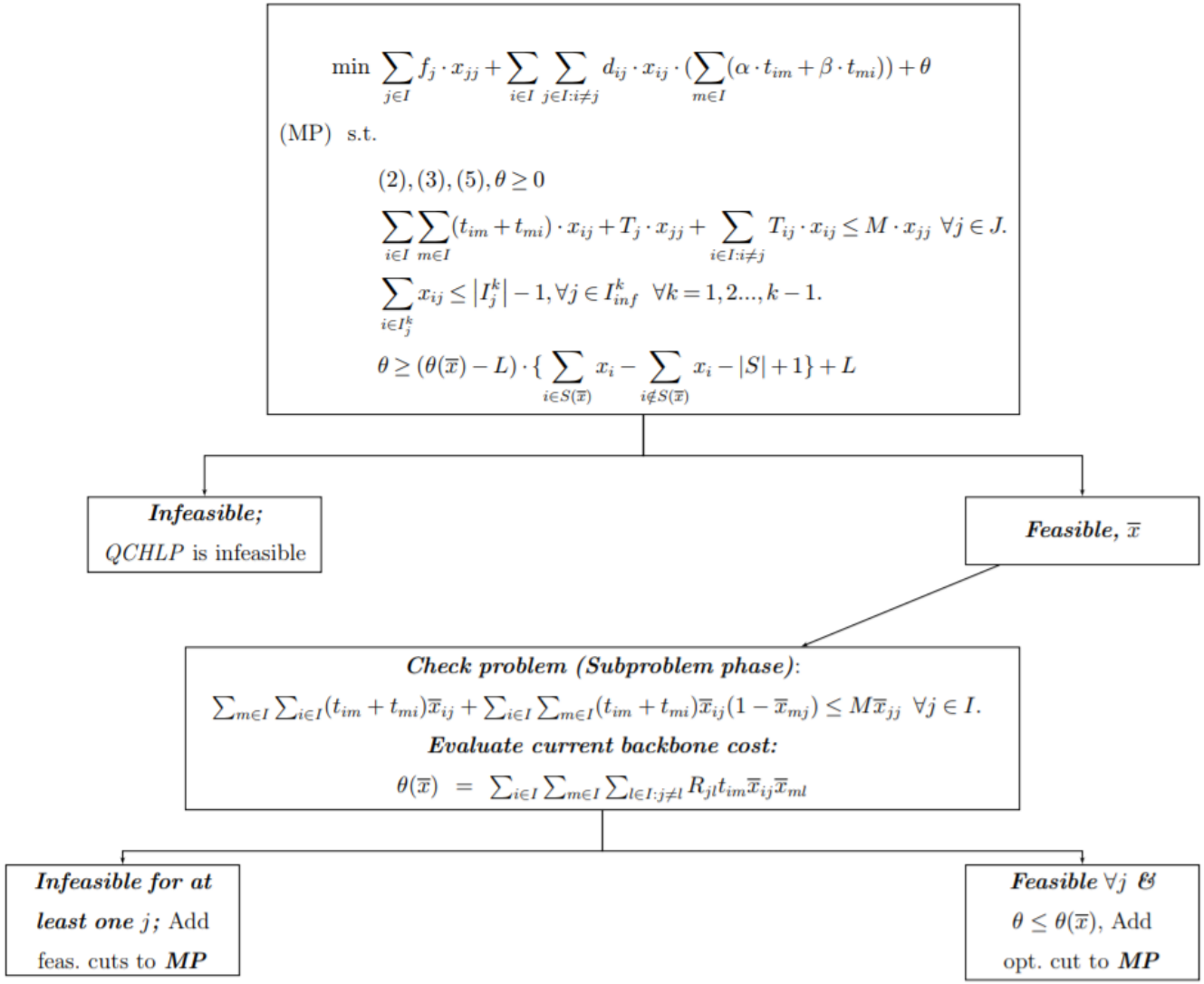


Figure 5.1: A Benders Decomposition Type Algorithm for Quadratic Capacitated Hub Location Problem

5.3 Computational Results for QCHLP

In this chapter, we develop an exact Benders decomposition type algorithm (BDTA) for QCHLP. In this decomposition algorithm, both feasibility and optimality cuts are generated to reach the optimal solution. Due to the structure of the problem, implementation of classical Benders decomposition is not possible. Therefore, we handle nonlinearities in both objective function and constraints without using any linearizations and LP duality theory. Our algorithm is based on logic based Benders decomposition in which generating cuts depends on the problem's structure.

In terms of the feasibility cuts, we propose two alternatives: MFC and EFC. As the relative performance of EFC is better than the one of MFC for QCCLP, we used EFC in our computational study. Similarly, there are two options for optimality cuts. In this section, the one that we proposed for QCHLP is compared with general purpose (*regular optimality cuts*) developed by Laporte and Louveaux (LL cuts.)

In the implementation of our Benders decomposition type algorithm, we used VI(2) as initial valid inequalities added to the MP. For obtaining lower bound values on the backbone links, the knapsack problems given in TB algorithm are solved as IP.

As QCHLP includes multiplications of binary variables in both objective function and constraints, it is also possible to solve it to optimum by using MIQCP solver of CPLEX. To find an optimal solution for such kind of problems, there are restrictions on the type of objective function and constraints. We refer the reader to A for more detailed information about solving mixed integer nonconvex problems optimally in IBM ILOG CPLEX.

By using the current advances in CPLEX, implementing automatic Benders for mixed integer linear programs is possible. To evaluate performance of BDTA with different optimality cuts, we compare decomposition algorithm with both MIQCP solver of CPLEX and automatic Benders implementation in IBM ILOG CPLEX 12.7.1.

In our computational study, we generated problem instances by using the AP data set in the literature. Except hub capacity values, all remaining parameters are set to the same values used for QCCLP. Since QCHLP includes additional routing cost in

the objective function, we generate new parameters, R_{jl}. In our study, R_{jl} values are determined as $R_{jl} = d_{jl} \cdot \gamma$, where γ represents the discount factor for one unit flow transferred between hub locations j and l. γ is the same in each hub link (j and l). In the AP data set, γ is given as 0.75. In order to see the effects of γ on overall performance of BDTA, we set two values for γ as 0.5 and 0.75.

To solve the QCHLP with MIQCP solver of CPLEX, we used default parameters.

As done in Chapter 4, we conduct our experimental setting on a 64-bit machine @ 3.10 GHz and 16 GB of Ram. All methods are coded in IBM CPLEX 12.7.1 using *C++ Concert technology*.

5.3.1 Computational Results for the instances with ($\gamma = 0.75$)

In this section, we first give computational results of our proposed Benders decomposition algorithm with MIQCP solver of CPLEX. Then in order to evaluate the performance of TBB cuts that we propose for QCHLP, we compare TBB optimality cuts with the general purpose optimality cuts (LL cuts).

- *Comparison with MIQCP solver in CPLEX and BDTA with TBB cuts*

  To solve the compact nonlinear integer program for QCHLP in CPLEX, we set the parameter for solving *MIQCPs* as *(IloCplex:Param:MIP:Strategy:MIQCPStrat, 2)*. All remaining parameters except the lift and project cuts in CPLEX are in their default values. For lift and project cuts, we used the following parameter setting *(MIP::Cuts::LiftProj:2)*.

  We give computational results for MIQCP solver of CPLEX and BDTA with TBB optimality cuts for 20 nodes in Table 5.4. The column 'Demand' describes the t_{im} values used in the models. In order to generate five instances, we divide the original demand values in AP data set by 2,3,4 and 5. For each experimental setting, in the first instance, demand values are set to original demand value in the literature. We may sometimes end up with infeasible instances. Therefore, we remove such kind of instances from analysis. For each alternative method, 'Obj' column gives the objective function value of the best integer solution

when time limit reached. When the method finds the optimal solution, these values reported in this column are the objective function values of the optimal solution. 'B& B' column for MIQCP gives the total number of open nodes in the branch-and-bound tree of the QCHLP. 'B&B' nodes column under the decomposition algorithm gives the number of nodes in B&B tree of the MP. The 'CPU' column for BDTA gives the total time including required time for calculating lower bound values on backbone traffic. In the 'Opt' row of the Table 5.4 gives the total number of instances that are solved to optimum by each method for given time limit, 7200 CPU seconds.

From Table 5.4, the first performance measure that we use to compare two methods *(MIQCP and BDTA with TBB optimality cuts)* is the average CPU time. Over 18 instances for 20-nodes, the average CPU time in *MIQCP* is 4,333.4 sec. while average CPU time is 1,692.9 sec. for BDTA. We could solve 14 instances to optimum over 18 instances, while MIQCP solver of CPLEX can solve 8 instances to optimum. When compared to the number of instances that are not solved to optimum by each method, BDTA gives a better solution than the solution of MIQCP. There are also instances (e.g, Instances 2,3,6,7,10 and 15 in Table 5.4) which are not optimally solved by MIQCP, while they could be solved to optimum by BDTA within given time limit. These results shows that BDTA is superior to the MIQCP solver for 20 nodes instances.

In Table 5.5, we give computational results for 25 nodes instances. CPLEX could solve MIQCP formulation to optimum in only one instance out of 16 instances. However, 12 instances are solved to optimum by using BDTA. One important result that can be seen in Table 5.5 is that the performance of MIQCP is not promising even for moderate size problems. Average relative gap for four instances which are not solved to optimum is 3 % in the BDTA. It is concluded that the performance of BDTA is better than the one of MIQCP. We can observe that solving 25 nodes instances is more difficult than the instances with 20 nodes. The performance of MIQCP solver becomes worse when the number of nodes in the network increases.

We give computational results for 40 nodes instances in Table 5.6. As can be seen from the results in Table 5.6, all generated instances for 40 nodes are

very challenging for CPLEX MIQCP solver. No instance over 18 instances is solved to optimum by CPLEX. However, BDTA could solve seven instances to optimum. When we look at the relative gap values in MIQCP, these values are high. The reason is that the lower bound in MIQCP is not good when time limit is reached. In CPU column of BDTA, we report CPU times higher than time limit. This result is due to adding the required time to calculate traffic bound values to the CPU time of BDTA. Note that, there are only two instances (Ins 17 and 18 From Table 5.6) for which MIQCP solver finds the optimal solution. However, the optimality is not proven within given time limit.

- *Comparison with BDTA with TBB optimality cuts and BDTA with LL optimality cuts*

  In order to see whether there is a significant difference between the optimality cuts (TBB optimality cuts vs LL optimality cuts), we compare BDTAs under different optimality cuts. Firstly, we will discuss the comparison results for the instances with $\gamma = 0.75$.

  We give computational results of comparing BDTA with TBB cuts and BDTA with LL cuts in Table 5.7. In terms of average CPU time over 18 instances, BDTA with TBB cuts gives smaller time than BDTA with LL cuts. BDTA, in which TBB cuts are used, is slightly better than the BDTA with LL cuts method in terms of optimally solved instances. BDTA with TBB cuts algorithm solves 14 instances to optimum, while BDTA with LL cuts solves 12 instances optimally. BDTA with LL cuts requires less number of feasibility cuts than BDTA with TBB cuts. However, average number of optimality cuts is higher in BDTA with LL cuts.

  In Table 5.8, the relative performance of BDTA with TBB cuts and BDTA with LL cuts is given. By using TBB cuts, we can solve more instances to optimum. Average CPU time in BDTA with TBB cuts is smaller than time in BDTA with LL cuts. From Table 5.8, BDTA with TBB cuts requires less number of optimality cuts. This implies problem specific optimality cuts demonstrates a better performance than general purpose optimality cuts (LL cuts).

  We give computational results of comparing BDTA by using two different op-

timality cuts for 40 nodes in Table 5.9. We generate 20 instances for 40 nodes. Since two instances are infeasible, we remove them from the analysis. Over 18 instances, both methods cannot solve most of the instances to optimum within given time limit. The reason is that the most challenging instances occur in 40-nodes. When compared the objective function values of the instances that are not solved to optimum by any method, BDTA with LL cuts finds a better solution in eight instances. This implies that when the optimal solution is not found within given time limit, BDTA with LL cuts demonstrates a better performance. However, in Table 5.9, there is one instance which is solved optimally by BDTA with TBB cuts, while it is solved to optimum by BDTA with LL cuts.

In this section, we compared a Benders decomposition algorithm with two different cuts and MIQCP solver of CPLEX with each other for 20, 25 and 40 nodes instances. We give summary results of the comparison of alternative methods in Table 5.3. We report average CPU time and the number of optimally solved instances in Table 5.3. From this survey table, it is concluded that BDTA with TBB cuts is significantly better than the remaining two methods in terms of average CPU time and the number of optimally solved instances for 20 and 25 nodes.

Table 5.3: Summary computational results for 20, 25 and 40 nodes with $\gamma = 0.75$

	MIQCP		***BDTA with TBB cuts***		***BDTA with LL cuts***	
Nodes	***Av. CPU***	***# Opt***	***Av. CPU***	***# Opt***	***Av. CPU***	***# Opt***
20	4,333.4	8/18	**1,692.9**	14/18	2,480.7	12/16
25	6,814.6	1/16	**1,902.5**	12/16	3,576.5	9/16
40	7,200.9	0/18	5,266.0	7/18	5,541.6	6/18

Table 5.4: Computational Results of MIQCP and BDTA with TBB optimality cuts for the instances with 20 nodes with $\gamma = 0.75$

					MIQCP					***BDTA with TBB Cuts***							
Inst	**Demand**	**Nodes**	**Cap**	**FC**	**Obj**	**CPU**	**B & B Nodes**	**R.Gap**	**Hubs**	**Obj**	**CPU**	**B & B Nodes**	**R.Gap**	**Hubs**	**Lazy**	**Fcut**	**OptCut**
1	D/2	20	T	L	227,267	7,200.2	14,451	0.59	10;11;13;14;20	206,460	7,234.8	891,908	0.07	1;7;11;14;18	7451	7909	3247
2	D/3				139,790	7,200.2	13,192	0.47	10;11;14	133,262	992.3	472,624	0	7;11;18	2310	2575	610
3	D/4				99,938	7,204.3	39,860	0.24	11;18	99,689	62.9	2,523	0	11;14	419	192	227
4	D/5				69,526	229.2	1,543	0	11	69,526	0.5	0	0	11	2	0	2
5	D/2	20	T	T	267,738	7,200.1	13,978	0.54	1;7;8;10;11	233,100	7,234.8	2,675,694	0.04	7;8;10;11;18	3354	5879	701
6	D/3				145,954	7,234.8	26,258	0.27	10;11;18	144,245	280.5	76,984	0	7;11;18	3135	2971	839
7	D/4				105,254	7,200.1	86,248	0.04	11;18	105,173	56.5	2,228	0	10;11	522	284	238
8	D/5				71,117	62.8	224	0	11	71,117	0.5	0	0	11	2	0	2
9	D	20	L	L	284,236	7,200.1	14,041	0.67	6;8;13;14;20	261,695	7,234.2	378,174	0.03	6;11;14;15	12349	6777	6232
10	D/2				143,768	7,200.4	27,668	0.66	7;15	140,665	37.1	566	0	7;14	74	0	74
11	D/3				100,072	3,275.5	42,012	0	10	100,072	0.1	0	0	10	2	0	2
12	D/4				81,408	1,489.7	12,058	0	10	81,408	27.9	0	0	10	2	0	2
13	D/5				69,526	186.6	1,646	0	11	69,526	0.5	0	0	11	2	0	2
14	D	20	L	T	336,930	7,200.3	8,750	0.66	5;8;10;17;18	303,999	7,234.2	564,857	0.014	7;11;18;19	17211	15884	2791
15	D/2				157,911	7,200.2	41,035	0.1	11;18	157,911	45.1	3,123	0	11;18	311	71	240
16	D/3				103,093	587.2	4,810	0	10	103,093	0.5	0	0	10	2	0	2
17	D/4				83,130	99.1	558	0	11	83,130	27.9	0	0	11	2	0	2
18	D/5				71,117	64.9	229	0	11	71,117	0.5	0	0	11	2	0	2
Average						4,333.4	19,364.5	0.2			**1,692.9**	281,593.4	0.009			2,363.4	845.3
OPT								8					**14**				

Table 5.5: Computational Results of MIQCP and BDTA with TBB optimality cuts for the instances for 25 nodes with $\gamma = 0.75$

					MIQCP					***BDTA with TBB Cuts***							
Ins	**Demand**	**Nodes**	**Cap**	**FC**	**Obj**	**CPU**	**B & B Nodes**	**R.Gap**	**Hubs**	**Obj**	**CPU**	**B & B Nodes**	**R.Gap**	**Hubs**	**Lazy**	**Fcut**	**OptCut**
1	D/2	25	T	L	238,246	7,202.1	4,991	0.90	1;12;13;16;23	202,626	7,264.2	521,049	0.11	8;11;14;23;24	10209	8908	4575
2	D/3				107,610	7,200.2	7,803	0.73	2;25	105,078	7,283.4	474,000	0.04	8;23;25	21667	38211	5993
3	D/4				101,257	7,200.1	4,067	0.74	9;18	96,823	104.1	3,584	0	9;24	394	124	270
4	D/5				70,785	7,200.2	12,342	0.42	13	70,785	27.6	0	0	13	2	0	2
5	D/2	25	T	T	350,577	7,200.2	5,607	0.71	6;14;15;21	281,475	7,264.2	824,593	0.12	9;10;11;14;24	9315	10993	2548
6	D/3				112,306	7,200.1	7,551	0.45	8;25	131,417	7,283.4	357,944	0.15	13;20;25	1714	20817	1533
7	D/4				123,227	7,200.2	6,115	0.38	11;14	123,227	119.9	6,341	0	11;24	1074	987	87
8	D/5				99,547	7,200.5	3,470	1.44	6;39	84,076	27.4	0	0	14	2	0	2
9	D/2	25	L	L	147,580	7,200.2	4,824	0.78	14;18	143,779	514.3	15,016	0	9;18	502	16	486
10	D/3				97,108	7,200.2	4,887	0.74	13;25	91,430	11.4	4,040	0	8;25	299	0	299
11	D/4				82,528	7,200.2	6,510	0.63	13	82,528	1.0	0	0	13	2	0	2
12	D/5				70,785	7,200.1	9,302	0.38	13	70,785	0.8	0	0	13	2	0	2
13	D/2	25	L	T	189,475	7,200.2	5,447	0.65	9;13	183,058	535.2	10,927	0	14;24	1103	615	488
14	D/3				94,393	7,200.2	11,427	0.25	25	94,393	1.0	0	0	25	2	0	2
15	D/4				97,422	7,200.2	12,388	0.03	14	97,422	1.0	0	0	14	2	0	2
16	D/5				84,076	1,028.2	2,189	0	14	84,076	0.7	0	0	14	2	0	2
Average						6,814.6	6,807.5	0.6			**1,902.5**	138,593.4	0.026			5,041.9	1,018.3
OPT								1					**12**				

Table 5.6: Computational Results of MIQCP and BDTA with TBB optimality cuts for the instances with 40 nodes ($\gamma = 0.75$)

					MIQCP					***BDTA with TBB Cuts***							
Inst	**Demand**	**Nodes**	**Cap**	**FC**	**Obj**	**CPU**	**B & B Nodes**	**R.Gap**	**Hubs**	**Obj**	**CPU**	**B & B Nodes**	**R.Gap**	**Hubs**	**Lazy**	**Fcut**	**OptCut**
1	D/2	40	T	L	267,832	7,200.6	358	1.01	2;7;11;22;27	207,196	9,029.1	260800	0.14	6;19;22;25;29	7903	15788	1243
2	D/3				163,176	7,200.7	7979	1.08	2;14;36;38	135,033	9,588.8	195500	0.09	14;19;28	16922	22704	924
3	D/4				113,649	7,201.1	4208	1.03	14;35	98,211	7,225.0	130992	0.02	14;29	15850	14798	1052
4	D/5				93,899	7,200.4	347	0.83	14;35	86,712	2,243.0	61124	0	14;19	1834	2	1832
5	D/2	40	T	T	367,126	7,200.5	1252	0.81	1;22;24;25;35;38	281,618	9,029.1	397566	0.13	14;19;21;22;35	6437	9540	889
6	D/3				179,296	7,201.2	3923	0.71	14;22;35	166,670	9,588.8	171009	0.02	14;19;22	16621	14674	3004
7	D/4				151,913	7,200.6	536	0.71	1;19;22	119,503	7,225.2	94520	0.012	14;22	17373	14963	2410
8	D/5				109,359	7,200.4	2684	0.66	14;19	106,025	7,202.2	90541	0.008	14;22	6512	2	6510
9	D	40	L	L	321,941	7,201.4	14456	1.62	2;24;28;34;38	269,922	8,732.4	215931	0.09	11;22;28;36	18281	25659	1441
10	D/2				152,633	7,201.9	5054	1.37	6;37	143,442	8,087.3	122022	0.02	14;19	4512	872	3640
11	D/3				155,294	7,201.4	4666	1.65	16;35;38	102,702	2.9	0	0	19	2	0	2
12	D/4				105,301	7,200.7	3809	0.99	6;27	82,543	3.2	0	0	19	2	0	2
13	D/5				77,400	7,200.7	382	0.8	29	70,447	3.0	0	0	19	2	0	2
14	D	40	L	T	466,146	7,200.6	11397	1.22	1;14;16;17;35;38	350,196	8732.4	308115	0.06	14;19;22;38	20587	30854	693
15	D/2				192,866	7,200.8	4881	1.05	19;22	175,242	8087.2	126432	0.015	14;19	15104	12776	2328
16	D/3				121,848	7,200.5	333	1.19	22	118,465	2.9	0	0	19	2	0	2
17	D/4				98,306	7,201.6	754	1.05	19	98,306	3.2	0	0	19	2	0	2
18	D/5				84,696	7,200.8	5048	0.99	22	84,696.4	3.0	0	0	22	2	0	2
Average						7,200.9	4,003.7	1.0			**5,266.0**	120,808.4	0.03			9,035.1	1,443.2
OPT								0					**7**				

Table 5.7: Computational Results of BDTA with TBB cuts and BDTA with LL cuts for 20 nodes for $\gamma = 0.75$

					BDTA with TBB Cuts								***BDTA with LL Cuts***							
Inst	**Demand**	**Nodes**	**Cap**	**FC**	**Obj**	**CPU**	**B & B Nodes**	**R.Gap**	**Hubs**	**Lazy**	**Fcut**	**OptCut**	**Obj**	**CPU**	**B & B Nodes**	**R.Gap**	**Hubs**	**Lazy**	**Fcut**	**OptCut**
1	D/2	20	T	L	206,460	7,234.8	891908	0.07	1;7;11;14;18	7451	7909	3247	200,895	7,234.8	931805	0.05	1;7;11;14;18	6688	5727	3953
2	D/3				133,262	992.3	472624	0	7;11;18	2310	2575	610	133,262	7,242.2	336552	0.025	7;11;18	14142	3086	11506
3	D/4				99,689	62.9	2523	0	11;14	419	192	227	99,689	550.6	17294	0	11;14	4320	799	3521
4	D/5				69,526	0.5	0	0	11	2	0	2	69,526	0.6	0	0	11	2	0	2
5	D/2	20	T	T	233,100	7,234.8	2675694	0.04	7;8;10;11;18	3354	5879	701	232,904	7,234.8	642307	0.06	1;8;10;11;18	8916	4873	5799
6	D/3				144,245	280.5	76984	0	7;11;18	3135	2971	839	144,245	7,242.2	114096	0.02	7;11;18	20332	4182	16413
7	D/4				105,173	56.5	2228	0	10;11	522	284	238	105,173	140.3	6572	0	10;11	1960	446	1514
8	D/5				71,117	0.5	0	0	11	2	0	2	71,117	0.5	0	0	11	2	0	2
9	D	20	L	L	261,695	7,234.2	378174	0.03	6;11;14;15	12349	6777	6232	261,695	7,234.3	152560	0.06	6;11;14;15	21656	4578	17316
10	D/2				140,665	37.1	566	0	7;14	74	0	74	140,665	166.7	13230	0	7;14	1990	22	1968
11	D/3				100,072	0.5	0	0	10	2	0	2	100,072	0.5	0	0	10	2	0	2
12	D/4				81,408	27.9	0	0	10	2	0	2	81,408	27.9	0	0	10	2	0	2
13	D/5				69,526	0.5	0	0	11	2	0	2	69,526	0.5	0	0	11	2	0	2
14	D	20	L	T	303,999	7,234.2	564857	0.014	7;11;18;19	17211	15884	2791	303,999	7,234.3	188972	0.05	7;11;18;19	22464	9416	13815
15	D/2				157,911	45.1	3123	0	11;18	311	71	240	157,911	313.0	16755	0	11;18	2852	104	2748
16	D/3				103,093	0.5	0	0	10	2	0	2	103,093	0.5	0	0	10	2	0	2
17	D/4				83,130	27.9	0	0	11	2	0	2	83,130	27.9	0	0	11	2	0	2
18	D/5				71,117	0.5	0	0	11	2	0	2	71,117	0.5	0	0	11	2	0	2
Average						**1,692.9**	281,593.4	0.009			2,363.4	845.3		2,480.7	134,452.4	0.015			1,846.3	4,364.9
OPT								**14**								12				

Table 5.8: Computational Results of BDTA with TBB cuts and BDTA with LL cuts for 25 nodes ($\gamma = 0.75$)

					BDTA with TBB Cuts								***BDTA with LL Cuts***							
Ins	**Demand**	**Nodes**	**Cap**	**FC**	**Obj**	**CPU**	**B & B Nodes**	**R.Gap**	**Hubs**	**Lazy**	**Fcut**	**OptCut**	**Obj**	**CPU**	**B & B Nodes**	**R.Gap**	**Hubs**	**Lazy**	**Fcut**	**OptCut**
1	D/2	25	T	L	202,626	7,264.2	521049	0.11	8;11;14;23;24	10209	8908	4575	203,830	7,264.2	265196	0.12	8;14;16;18;24	13737	5856	9529
2	D/3				105,078	7,283.4	474000	0.04	8;23;25	21667	38211	5993	104,967	7,283.6	161172	0.05	8;23;25	14485	9007	8712
3	D/4				96,823	104.1	3584	0.00	9;24	394	124	270	96,823	4,421.6	48873	0.00	9;24	9667	1755	7912
4	D/5				70,785	27.6	0	0.00	13	2	0	2	70,785	27.4	0	0.00	13	2	0	2
5	D/2	25	T	T	281,475	7,264.2	824593	0.12	9;10;11;14;24	9315	10993	2548	279,109	7,264.2	320505	0.12	9;11;14;21;24	14051	7514	8365
6	D/3				131,417	7,283.4	357944	0.15	13;20;25	1714	20817	1533	129,304	7,283.4	317710	0.14	13;20;25	14650	16408	2519
7	D/4				123,227	119.9	6341	0.00	11;24	1074	987	87	123,227	1,121.5	17199	0.00	11;14	5813	2310	3503
8	D/5				84,076	27.4	0	0.00	14	2	0	2	84,076	27.4	0	0.00	14	2	0	2
9	D/2	25	L	L	143,779	514.3	15016	0.00	9;18	502	16	486	143,779	7,662.3	165292	0.03	9;18	11541	150	11391
10	D/3				91,430	11.4	4040	0.00	8;25	299	0	299	91,430	7,201.2	98707	0.03	8;25	11192	0	11192
11	D/4				82,528	1.0	0	0.00	13	2	0	2	82,528	1.0	0	0.00	13	2	0	2
12	D/5				70,785	0.8	0	0.00	13	2	0	2	70,785	0.7	0	0.00	13	2	0	2
13	D/2	25	L	T	183,058	535.2	10927	0.00	14;24	1103	615	488	183,058	7,662.4	105591	0.02	14;24	12059	1078	10981
14	D/3				94,393	1.0	0	0.00	25	2	0	2	94,393	1.0	0	0.00	25	2	0	2
15	D/4				97,422	1.0	0	0.00	14	2	0	2	97,422	1.0	0	0.00	14	2	0	2
16	D/5				84,076	0.7	0	0.00	14	2	0	2	84,076	0.7	0	0.00	14	2	0	2
Average						**1,902.5**	138,593.4	0.0		2,893.2	5,041.9	1,018.3		3,576.5	93,765.3	0.0		6,700.6	2,754.9	4,632.4
OPT								**12**								9				

Table 5.9: Computational Results of BDTA with TBB cuts and BDTA with LL cuts for 40 nodes-instances with $\gamma = 0.75$

					BDTA with TBB Cuts								***BDTA with LL Cuts***							
Ins	**Demand**	**Nodes**	**Cap**	**FC**	**Obj**	**CPU**	**B & B Nodes**	**R.Gap**	**Hubs**	**Lazy**	**Fcut**	**OptCut**	**Obj**	**CPU**	**B & B Nodes**	**R.Gap**	**Hubs**	**Lazy**	**Fcut**	**OptCut**
1	D/2	40	T	L	207,196	9,029.1	260800	0.14	6;19;22;25;29	7903	15788	1243	208,975	9,029.4	126799	0.157	6;22;25;28;38	9951	8967	4686
2	D/3				135,033	9,588.8	195500	0.09	14;19;28	16922	22704	924	201,428	9,588.9	159600	0.127	6;19;22;25;29	6272	7714	2702
3	D/4				98,211	7,225.0	130992	0.02	14;29	15850	14798	1052	98,211	7,225.3	60523	0.03	14;29	9667	5025	4642
4	D/5				86,712	**2,243.0**	61124	0	14;19	1834	2	1832	87,480	7,201.8	93895	0.02	14;19	5875	1	5874
5	D/2	40	T	T	281,618	9,029.1	397566	0.13	14;19;21;22;35	6437	9540	889	277,360	9,029.1	313700	0.12	14;19;21;22;35	3485	3134	1507
6	D/3				166,670	9,588.8	171009	0.02	14;19;22	16621	14674	3004	164,996	9,588.9	52814	0.015	14;19;22	13527	7696	6049
7	D/4				119,503	7,225.2	94520	0.012	14;22	17373	14963	2410	118,841	7,225.1	48995	0.007	14;22	15317	11241	4076
8	D/5				106,025	7,202.2	90541	0.008	14;22	6512	2	6510	105,573	7,202.9	87411	0.005	14;22	6381	3	6378
9	D	40	L	L	269,922	8,732.4	215931	0.09	11;22;28;36	18281	25659	1441	259,748	8,732.5	123600	0.06	11;22;29;36	13910	16480	2109
10	D/2				143,442	8,087.3	122022	0.02	14;19	4512	872	3640	140,604	8,087.8	69757	0.03	14;29	6712	297	6415
11	D/3				102,702	2.9	0	0	19	2	0	2	102,702	2.9	0	0	19	2	0	2
12	D/4				82,543	3.2	0	0	19	2	0	2	82,543	3.2	0	0	19	2	0	2
13	D/5				70,447	3.0	0	0	19	2	0	2	70,447	3.0	0	0	19	2	0	2
14	D	40	L	T	350,196	8,732.4	308115	0.06	14;19;22;38	20587	30854	693	341,578	8,732.4	99313	0.05	14;19;22;35	19787	21468	2810
15	D/2				175,242	8,087.2	126432	0.015	14;19	15104	12776	2328	173,290	8,087.4	71427	0.01	14;19	9846	4137	5709
16	D/3				118,465	2.9	0	0	19	2	0	2	118,465	2.9	0	0	19	2	0	2
17	D/4				98,306	3.2	0	0	19	2	0	2	98,306	3.2	0	0	19	2	0	2
18	D/5				84,696.4	3.0	0	0	22	2	0	2	84,696	3.0	0	0	22	2	0	2
Average						**5,266.0**	120,808.4	0.0			9,035.1	1,443.2		5,541.6	72,657.4	0.0			4,786.8	2,942.7
OPT								7								6				

5.3.2 Computational Results for the instances with ($\gamma = 0.50$)

In this section, we first compare the relative performance of BDTA in which TBB cuts are used with MIQCP solver of CPLEX for 20, 25 and 40 nodes.

We give computational results of MIQCP and BDTA with TBB cuts for 20 nodes instances where $\gamma = 0.50$ in Table 5.10. BDTA solved 17 instances over 18 instances to optimum. In the instance that is not solved optimally by BDTA, the relative gap is 3%. However, the solutions found in MIQCP are worse than the solutions in BDTA with TBB cuts. For MIQCP, there are some instances in which the optimal hub locations and the assignments are chosen but the optimality is not proven within given time limit. These instances (2,3,10,11 and 14) are given in 5.10. From this table, we conclude that BDTA is better than MIQCP solver in terms of average CPU time. Even for the instances where the optimal number of hubs is one, MIQCP solver requires much time than BDTA.

In Table 5.11, we give computational results of comparing MIQCP and BDTA with TBB cuts for 25 nodes. We can conclude that BDTA is significantly better than MIQCP solver in CPLEX in terms of number of optimally solved instances and average CPU time. BDTA solves eight more instances to optimum, while they are not solved by using MIQCP. For the instance 5, which is not optimally solved by any method, BDTA found a better solution than MIQCP.

Table 5.12 gives the comparison results of BDTA with different optimality cuts: *TBB cuts and LL cuts*. By using TBB cuts, BDTA requires less amount of computational time than BDTA with LL cuts. As we expected, average number of optimality cuts in BDTA with TBB is less than the average number of cuts in BDTA with LL cuts. Since more instances are solved to optimum by using BDTA with TBB cuts, we can say that the performance of BDTA with TBB cuts is better than the one of BDTA with LL cuts.

As no instance in 40 nodes could be solved to optimality by using MIQCP solver of CPLEX, we only report the results for BDTA with TBB cuts and BDTA with LL cuts in Table 5.14. Most of the instances couldn't be solved to optimum by using any method. However, in Table 5.14, there are instances that are not optimally solved within given time limit by using BDTA with LL cuts. For these instances, BDTA with TBB cuts finds the optimal solutions. On the

other hand, BDTA with LL cuts can find a better solution than the algorithm using TBB cuts for the instances (Ins 5,8,11,12) that are not solved to optimum within 7200 CPU seconds time limit.

In Table 5.15, we report the summary results of comparing MIQCP, BDTA with TBB cuts and BDTA with LL cuts for different number of nodes in the network under two discount factor values between hubs: 0.75 and 0.5. When we decrease the discount factor from 0.75 to 0.5, it is concluded that average CPU time decreases and the number of instances solved to optimum increases in each alternative method for 20 and 25 nodes. In the last row of Table 5.15, we report % reduction in CPU time when γ is decreased from 0.75 to 0.5. The maximum CPU time reduction by changing the discount factor value is observed in 20-nodes instances by using the BDTA with TBB cuts.

In this section, we also compare the relative performance of automatic Benders decomposition (*AutoBEND*) implementation with the one of BDTA. To apply automatic Benders decomposition algorithm, we first linearize the nonlinear integer model of QCHLP. Then, the default configuration for this implementation is chosen in IBM ILOG CPLEX and fullAutomatic Benders decomposition is employed.

As *AutoBEND* decomposition cannot find an integer feasible solution for 40 nodes instances, we only give the computational comparison of branch-and-check and *AutoBEND* for 20 and 25 nodes. The reason why *AutoBEND* cannot find an integer feasible solution is that when the number of nodes in the networks is high, the required number of decision variables in the linearization increases dramatically.

We give computational results of comparing BDTA with automatic Benders decomposition in Tables 5.16 and 5.17 for 20 nodes and 25 nodes, respectively. From Table 5.16, we conclude that BDTA with TBB cuts outperforms the AutoBend implementation in CPLEX. All instances are solved to optimum by BDTA. However, four instances couldn't be solved optimally by *AutoBend* within given time limit. For each instance given in Table 5.16, BDTA requires less computational time than AutoBend. Moreover, when demand values are high, the solutions found by *AutoBend* are worse than those in BDTA.

From Table 5.17, we can observe that when the number of nodes increases from 20 to 25, the performance of *AutoBend* implementation becomes better for some outlier instances. Consider the instance 2 given in Table 5.17. This instance can be solved by implementing AutoBend in CPLEX after linearizing the QCHLP. However, BDTA with TBB cuts reaches the time limit without finding the optimal solution for this instance. In Table 5.17, there are two instances (Ins 1 and Ins 5) which are not solved to optimum by any alternative method within given time limit. For these instances, the solutions found in AutoBend implementation is worse than those obtained in BDTA. This result can also be observed from the relative gap values in Table 5.17. One observation that can be conducted from Table 5.17 is *AutoBend* implementation can require higher CPU times for the instances that can be solved by BDTA at the root node.

Average CPU time in *AutoBend* implementation is less than the time required for BDTA. We believe that this result is due to the fact that total required time to calculate lower bounds for traffic bound is high for 25 nodes instances. To increase the convergence speed of BDTA, the strategy that is solving knapsack problems as LP could be used.

Table 5.10: Computational Results of MIQCP and BDTA with TBB cuts for 20 nodes instances ($\gamma = 0.50$)

					MIQCP					***BDTA with TBB cuts***							
Demand	**Ins**	**Nodes**	**Cap**	**FC**	**Obj**	**CPU**	**B & B Nodes**	**R.Gap**	**Hubs**	**Obj**	**CPU**	**B & B Nodes**	**R.Gap**	**Hubs**	**Lazy**	**Fcut**	**OptCut**
D/2	1	20	T	L	210,072	7,235	17315	0,43	**9;10;11;14;19**	200,895	7,234.8	2714588	0,03	**1;7;11;14;18**	3521	6898	549
D/3	2				130,367	7,242	22550	0,26	**7;11;18**	130,367	182.0	177823	0	**7;11;18**	475	623	97
D/4	3				98,023	5,212	36808	0	**11;18**	98,023	50.2	572	0	**11;18**	131	52	79
D/5	4				69,526	57	190	0	**11**	69,526	0.6	0	0	**11**	2	0	2
D/2	5	20	T	T	241,485	7,235	20658	0,35	**1;8;10;11;18**	226,298	2,850.6	2211710	0	**1;8;10;11;18**	1779	4020	201
D/3	6				142,046	7,242	30501	0,18	**1;11;18**	141,351	74.5	8790	0	**7;11;18**	423	445	108
D/4	7				103,579	829	13010	0	**11;18**	103,579	45.6	151	0	**11;18**	44	11	33
D/5	8				71,117	30	84	0	**11**	71,117	0.6	0	0	**11**	2	0	2
D	9	20	L	L	277,379	7,234	18684	0,42	**2;9;11;14;15**	253,629	331.8	66121	0	**6;11;14;19**	2343	1626	999
D/2	10				137,507	7,237	35180	0,38	**7;14**	137,507	36.6	101	0	**7;14**	28	0	28
D/3	11				100,072	7,200	65213	0,19	**10**	100,072	0.5	0	0	**10**	2	0	2
D/4	12				81,408	2,262	36801	0	**10**	81,408	28.0	0	0	**10**	2	0	2
D/5	13				69,526	142	2483	0	**10**	69,526	0.6	0	0	**10**	2	0	2
D	14	20	L	T	295,168	7,234	211161	0,01	**7;11;18;19**	295,168	931.9	137479	0	**7;11;18;19**	7996	7844	1020
D/2	15				154,560	1,632	12296	0	**11;18**	154,560	38.2	303	0	**11;18**	54	6	48
D/3	16				103,093	71	153	0	**10**	103,093	0.5	0	0	**10**	2	0	2
D/4	17				83,130	111	897	0	**11**	83,130	28.0	0	0	**11**	2	0	2
D/5	18				71,117	27	140	0	**11**	71,117	0.6	0	0	**11**	2	0	2
Average						3,791	29,118	0,12			**658**	295,424	0,002			1,196	177
Opt								10					**17**				

Table 5.11: Computational Results of MIQCP and BDTA with TBB cuts for 25 instances ($\gamma = 0.50$)

					MIQCP					***BDTA with TBB cuts***							
Demand	**Ins**	**Nodes**	**Cap**	**FC**	**Obj**	**CPU**	**B & B Nodes**	**R.Gap**	**Hubs**	**Obj**	**CPU**	**B & B Nodes**	**R.Gap**	**Hubs**	**Lazy**	**Fcut**	**OptCut**
D/2	1	25	T	L	216,877	7,264.5	4418	0.64	**10;11;13;24**	197,389	7,264.2	1109000	0.08	**8;11;14;23;24**	4848	6648	1782
D/3	2				96,768	7,283.8	19728	0.49	**4;25**	102,249	7,283.4	450143	0.05	**8;23;25**	25102	49705	136
D/4	3				97,642	7,285.3	4660	0.5	**14;24**	95,014.3	96.5	553	0	**9;24**	85	31	54
D/5	4				70,785	7,227.7	28583	0.22	**13**	70,785	27.6	0	0	**13**	2	0	2
D/2	5	25	T	T	325,411	7,264.3	7097	0.55	**9;12;14;15;21**	272,151	7,264.2	1497971	0.09	**4;9;11;14;21;24**	5606	8787	993
D/4	6				121,241	7,285.3	9056	0.14	**11;14**	121,241	108.8	721	0	**11;14**	188	163	25
D/5	7				84,076	237.4	868	0	**14**	84,076	27.6	0	0	**14**	2	0	2
D/2	8	25	L	L	141,431	7,662.7	22983	0.64	**8;18**	140,701	482.9	1028	0	**9;18**	159	0	159
D/3	9				88,882	7,202.2	20649	0.54	**8;25**	88,882	8.2	442	0	**8;25**	61	0	61
D/4	10				82,528	7,201.1	12864	0.16	**13**	82,528	1.2	0	0	**13**	2	0	2
D/5	11				70,785	7,200.8	29100	0.22	**13**	70,785	0.9	0	0	**13**	2	0	2
D/2	12	25	L	T	185,452	7,662.5	15630	0.43	**11;13**	179,606	486.6	1101	0	**11;14**	185	74	111
D/3	13				141,948	7,203.3	22889	0.64	**8;18**	140,701	20.2	1028	0	**9,18**	60	0	60
D/4	14				97,422	180.1	201	0	**14**	97,422	1.0	0	0	**14**	2	0	2
D/5	15				84,076	248.2	960	0	**14**	84,076	0.9	0	0	**14**	2	0	2
Average						5,894	13,312	0.34			**1,538**	204,132	0			4,361	226
Opt								3					**12**				

Table 5.12: Comparison results of BDTA with TBB cuts and BDTA with LL cuts for 20 nodes instances with $\gamma = 0.50$

					BDTA with TBB cuts								***BDTA with LL cuts***							
Ins	**Demand**	**Nodes**	**Cap**	**FC**	**Obj**	**CPU**	**B & B Nodes**	**R.Gap**	**Hubs**	**Lazy**	**Fcut**	**OptCut**	**Obj**	**CPU**	**B & B Nodes**	**R.Gap**	**Hubs**	**Lazy**	**Fcut**	**OptCut**
1	D/3	20	T	L	130,367	182.0	177823	0	**7;11;18**	475	623	97	130,367	1,450.5	168070	0	**7;11;18**	5532	1230	4513
2	D/4				98,023	50.2	572	0	**11;18**	131	52	79	98,023	69.9	4280	0	**11;18**	865	142	723
3	D/5				69,526	0.6	0	0	**11**	2	0	2	69,526	0.5	0	0	**11**	2	0	2
4	D/2	20	T	T	226,298	2,850.6	2211710	0	**1;8;10;11;18**	1,779	4020	201	226,298	7,234.8	1046051	0.02	**1;8;10;11;18**	6436	3701	4546
5	D/3				141,351	74.5	8790	0	**7;11;18**	423	445	108	141,351	932.6	27933	0	**7;11;18**	6779	1487	5439
6	D/4				103,579	45.6	151	0	**11;18**	44	11	33	103,579	50.0	1222	0	**11;18**	330	46	284
7	D/5				71,117	0.6	0	0	**11**	2	0	2	71,117	0.6	0	0	**11**	2	0	2
8	D	20	L	L	253,629	331.8	66121	0	**6;11;14;19**	2,343	1626	999	253,629	7,234.4	204941	0.02	**6;11;14;19**	18335	2684	15795
9	D/2				137,507	36.6	101	0	**7;14**	28	0	28	137,507	43.1	2215	0	**7;14**	383	1	382
10	D/3				100,072	0.5	0	0	**10**	2	0	2	100,072	0.5	0	0	**10**	2	0	2
11	D/4				81,408	28.0	0	0	**10**	2	0	2	81,408	27.9	0	0	**10**	2	0	2
12	D/5				69,526	0.6	0	0	**10**	2	0	2	69,526	0.6	0	0	**10**	2	0	2
13	D/2	20	L	T	154,560	38.2	303	0	**11;18**	54	6	48	154,560	50.4	3509	0	**11;18**	536	7	529
14	D/3				103,093	0.5	0	0	**10**	2	0	2	103,093	0.5	0	0	**10**	2	0	2
15	D/4				82,130	28.0	0	0	**10**	2	0	2	82,130	27.9	0	0	**10**	2	0	2
16	D/5				71,117	0.6	0	0	**10**	2	0	2	71,117	0.5	0	0	**10**	2	0	2
Average						**229.3**		0		330.8	**423.9**	**100.6**		1,070.3		0.0		2,450.8	581.1	2,014.2
Opt						**16**								14						

Table 5.13: Comparison Results of BDTA with TBB cuts and BDTA with LL cuts for 25 nodes with $\gamma = 0.50$

					BDTA with TBB cuts								***BDTA with LL cuts***							
Ins	**Demand**	**Nodes**	**Cap**	**FC**	**Obj**	**CPU**	**B & B Nodes**	**R.Gap**	**Hubs**	**Lazy**	**Fcut**	**OptCut**	**Obj**	**CPU**	**B & B Nodes**	**Sts**	**R.Gap**	**Hubs**	**Lazy**	**Fcut**	**OptCut**
1	D/2	25	T	L	197,389	7,264.2	1109000	0.08	**8;11;14;23;24**	4848	6648	1782	197,389	7,264.2	531686	ATL	0.09	**8;11;14;23;24**	8597	5667	5565
2	D/3				102,249	7,283.4	450143	0.05	**8;23;25**	25102	49705	136	102,249	7,283.4	486064	ATL	0.05	**8;23;25**	25196	48985	525
3	D/4				95,014	96.5	553	0	**9;24**	85	31	54	95,104	234.0	7251	Opt	0	**9;24**	1721	365	1356
4	D/5				70,785	27.6	0	0	**13**	2	0	2	70,785	27.4	0	Opt	0	**13**	2	0	2
5	D/2	25	T	T	272,151	7,264.2	1497971	0.09	**4;9;11;14;21;24**	5606	8787	993	272,151	7,264.2	519924	ATL	0.09	**4;9;11;14;21;24**	7302	5102	4120
6	D/4				121,241	108.8	721	0	**11;14**	188	163	25	121,241	118.4	2575	Opt	0	**11;14**	805	389	416
7	D/5				84,076	27.6	0	0	**14**	2	0	2	84,076	27.4	0	Opt	0	**14**	2	0	2
8	D/2	25	L	L	140,701	482.9	1028	0	**9;18**	159	0	159	140,701	4,258.8	93154	OptTol	0	**9;18**	6803	40	6763
9	D/3				88,882	8.2	442	0	**8;25**	61	0	61	88,882	958.5	24145	Opt	0	**8;25**	3821	0	3821
10	D/4				82,528	1.2	0	0	**13**	2	0	2	82,528	1.0	0	Opt	0	**13**	2	0	2
11	D/5				70,785	0.9	0	0	**13**	2	0	2	70,785	0.7	0	Opt	0	**13**	2	0	2
12	D	25	L	T	179,606	24.5	1101	0	**11;14**	185	74	111	179,606	732.1	20281	Opt	0	**11;14**	3500	296	3204
13	D/2				140,701	482.8	1028	0	**14**	159	0	159	-	-	-	-	-	**-**	-	-	-
15	D/5				84,076	0.2	0	0	**14**	2	0	2	84,076	0.1	0	Opt	0	**14**	2	0	2
Average						**1,648.1**	218,713.4	0.0		2,600.2	4,672.0	249.3		2,166.9	129,621.5				4,442.7	4,680.3	1,983.1
Opt								11										10			

Table 5.14: Comparison of BDTA with TBB cuts and BDTA with LL cuts for 40 nodes with $\gamma = 0.5$

					BDTA with TBB Cuts								***BDTA with LL Cuts***							
Inst	**Demand**	**Nodes**	**Cap**	**FC**	**Obj**	**CPU**	**B & B Nodes**	**R.Gap**	**Hubs**	**Lazy**	**Fcut**	**OptCut**	**Obj**	**CPU**	**B & B Nodes**	**R.Gap**	**Hubs**	**Lazy**	**Fcut**	**OptCut**
1	D/2	40	T	L	200,295	9,029.1	364260	0.11	**6;19;22;29;35**	6789	14686	951	201,428	9,029.0	317807	0.12	**6;19;22;25;29**	6626	11050	2345
2	D/3				131,182	9,588.8	317930	0.05	**14;19;29**	9734	12905	453	132,386	9,588.8	131495	0.07	**14;19;38**	10133	10678	1905
3	D/4				96,726	**968.8**	52452	0	**14;29**	7851	7527	324	96,726	7,225.2	70600	0.02	**14,29**	9935	5329	4606
4	D/5				86,712	**2,059.4**	61124	0	**14;19**	1834	2	1832	86,712	7,201.8	72584	0.01	**14;19**	5967	2	5965
5	D/2	40	T	T	274,368	9,029.1	1512400	0.09	**6;14;19;22;35**	3347	6297	274	265,710	9,029.1	417710	0.06	**1;14;19;22;35**	13958	23657	1074
6	D/3				164,996	9,588.9	246237	0.01	**14;19;22**	17152	15079	3210	164,996	9,588.8	80114	0.02	**14;19;22**	12346	6149	6378
7	D/5				105,573	7,201.9	116977	0.00	**14;22**	5735	2	5733	105,573	7,201.8	87006	0.01	**14;22**	6360	3	6357
8	D/2	40	L	L	260,216	8,087.3	265161	0.05	**11;22;28;36**	19560	26228	1547	259,748	8,087.3	145700	0.06	**11;22;29;36**	16048	17859	2523
9	D/3				140,604	**664.4**	26127	0	**14;29**	1324	294	1030	140,604	7,203.4	65670	0.03	**14;29**	6419	250	6169
10	D/4				102,702	3.0	0	0	**19**	2	0	2	102,702	3.2	0	0	**19**	2	0	2
11	D/2	40	L	T	342,796	8,087.2	259410	0.05	**14;19;22;35**	25799	37523	848	341,578	8,087.2	95666	0.05	**14;19,22;35**	19568	21600	2613
12	D/3				342,976	7,202.6	262824	0.05	**14;19;22;35**	25691	37724	866	341,578	7,202.6	95666	0.05	**14;19,22;35**	19568	21600	2613
13	D/4				118,465	3.0	0	0	**19**	2	0	2	118,465	3.2	0	0	**19**	2	0	2
Average						**5,624.2**	257,812.2	0.0			12,463.3	1,411.6		6,880.9	121,539.8	0.0			**9,090.5**	3,273.2
Opt								5								2				

Table 5.15: Comparison of MIQCP, BDTA with TBB cuts and BDTA with LL cuts in terms of CPU time and no of instances solved to optimum

Discount factor	***Methods***	*MIQCP*			*BDTA with TBB cuts*			*BDTA with LL cuts*		
	Performance Meas.	***20 Nodes***	***25 Nodes***	***40 Nodes***	***20 Nodes***	***25 Nodes***	***40 Nodes***	***20 Nodes***	***25 Nodes***	***40 Nodes***
0.75	*CPU time*	4,333.39	6,814.57	7,200.87	1,668.52	1,811.9	4,524.71	2,177.3	3,485.9	4,800.31
	No of inst. solved	8/18	1/16	0/18	14 /18	12 /16	7/18	12 /18	9/16	6/18
0.5	*CPU time*	3,337.17	6,204.6	7,200	206.23	550.57	5,624.2	1,047.22	2,097.6	6,880.9
	No of inst. solved	10/18	2 /16	0/13	16/18	12/16	5/13	15/18	10/16	2/13
% Reduction in CPU time		22.99	8.95		87.64	69.61		51.9	39.83	

Table 5.16: Computational Results of BDTA and AutoBend implementation in CPLEX for 20 nodes instances $\gamma = 0.50$

					BDTA with TBB cuts				***Linearization + AutoBend***			
Ins	**Demand**	**Nodes**	**Cap**	**FC**	**Obj**	**CPU**	**B&B Nodes**	**R.Gap**	**Obj**	**CPU**	**B&B Nodes**	**R.Gap**
1	D/3	20	T	L	130,367	**182.0**	177823	0	130,367	7,200.7	91376	0.07
2	D/4				98,023	**50.2**	572	0	98,023	95.3	325	0
3	D/5				69,526	**0.6**	0	0	69,526	42.8	0	0
4	D/2	20	T	T	226,298	**2,850.6**	2211710	0	266,554	7,201.0	27586	0.34
5	D/3				141,351	**74.5**	8790	0	141,351	7,200.8	171540	0.06
6	D/4				103,579	**45.6**	151	0	103,579	77.2	61	0
7	D/5				71,117	**0.6**	0	0	71,117	41.1	0	0
8	D	20	L	L	253,629	**331.8**	66121	0	263,553	7,200.7	57763	0.09
9	D/2				137,507	**36.6**	101	0	137,507	55.6	19	0
10	D/3				100,072	**0.5**	0	0	100,072	41.4	0	0
11	D/4				81,408	**28.0**	0	0	81,408	42.0	0	0
12	D/5				69,526	**0.6**	0	0	69,526	40.5	0	0
13	D/2	20	L	T	154,560	**38.2**	303	0	154,560	90.9	112	0
14	D/3				103,093	**0.5**	0	0	103,093	42.3	0	0
15	D/4				82,130	**28.0**	0	0	82,130	38.8	0	0
16	D/5				71,117	**0.6**	0	0	71,117	39.6	0	0
Average						**229.3**		**0**		**1,840.7**		**0.04**
Sum								**16**				12

Table 5.17: Computational results of BDTA with TBB cuts and AutoBend implementation in CPLEX for 25 nodes instances $\gamma = 0.50$

					BDTA with TBB cuts				***Linearizations + AutoBEND***			
Ins	**Demand**	**Nodes**	**Cap**	**FC**	**Obj**	**CPU**	**B&B Nodes**	**R.Gap**	**Obj**	**CPU**	**B&B Nodes**	**R.Gap**
1	D/2	25	T	L	197,389	7,264.2	1,109,000	0.08	227,381	7,203.2	13,885	0.32
2	D/3				102,249	7,283.4	450,143	0.05	88,967	184.64	9	0
3	D/4				95,104	96.5	553	0	95,104	241.78	176	0
4	D/5				70,785	27.6	0	0	70,785	120.67	0	0
5	D/2	25	T	L	272,151	7,264.2	1,497,971	0.09	534,942	7,202.7	10,528	0.63
6	D/4				121,241	27.6	721	0	121,241	272.77	109	0
7	D/5				84,709	0.25	0	0	84,709	122.59	0	0
8	D/2	25	L	L	140,701	482.9	1028	0	140,701	152.34	32	0
9	D/3				88,882	8.2	442	0	88,882	164.66	13	0
10	D/4				82,528	1.2	0	0	82,528	115.13	0	0
11	D/5				70,785	0.9	0	0	70,785	120.92	0	0
12	D	25	L	T	179,606	24.5	1101	0	179,606	201.16	96	0
13	D/2				140,401	482.8	1028	0	140,701	156.91	32	0
14	D/5				84,076	0.23	0	0	84,076	122.79	0	0
Average						1,640.3	218,713.4	0.02		1,172.7	1,177.1	0.07
Opt								11				12

5.3.3 Linear Capacitated Quadratic Hub Location Problem

So far, we evaluated the performance of our proposed Benders decomposition type algorithm for QCHLP. Note that, the decomposition algorithm for QCHLP includes both feasibility and optimality cuts. When compared BDTA under different optimality cuts (*TBB cuts and LL cuts*), we cannot observe the effect of using *problem specific optimality cuts* explicitly due to the existence of feasibility cuts.

In order to assess the effects of using problem specific optimality cuts rather than *LL cuts*, we consider a linear capacitated quadratic hub location problem (LCQHLP). This problem is also mentioned in [53], which is a special case of QCHLP. The nonlinear capacity constraints in QCHLP are replaced with linear constraints. The mathematical model for this problem ([53]) is given as follows:

$$\min \sum_{j\in I} f_j x_{jj} + \sum_{i\in I}\sum_{j\in I:i\neq j} d_{ij}x_{ij}(\sum_{m\in I}(\alpha t_{im} + \beta t_{mi}))+ \sum_{i\in I}\sum_{j\in I}\sum_{m\in I}\sum_{l\in I:j\neq l} R_{jl}t_{im}x_{ij}x_{ml} \tag{5.19}$$

(LCQLP) s.t.

$$(5.2),\ \ (5.3),\ \ (5.5)$$

$$\sum_{i\in I}\sum_{m\in I}(t_{im} + t_{mi})x_{ij} \leq Mx_{jj}, \forall j \in I. \tag{5.20}$$

Although routing cost related to backbone traffic in the objective function of *LCHLP*, backbone traffic does not use the capacity of the hubs. All remaining constraints (*assignment* (5.2), *logic relation* (5.3), and *sign restrictions* (5.5)) are the same as the constraints in QCHLP.

It is also possible to solve LCQHLP to optimum by using commercial solvers (e.g., CPLEX). As the objective function is nonlinear and the capacity constraints are linear, LCQHLP is in the class of quadratic programming. To solve such kind of optimization problems to optimum in CPLEX, we use quadratic programming solver of CPLEX with related parameters.

In the following section, we give Benders decomposition algorithm for LCQHLP, which is a special case of the decomposition algorithm for QCHLP.

5.3.3.1 Benders Decomposition Type Algorithm for LCQHLP

The same idea behind the decomposition that we used for *QCHLP* is also valid for *LCQHLP*. In the master problem of the decomposition for LCQHLP, we include linear capacity constraints. Therefore, the solution obtained from MP satisfies the capacity restrictions on the hubs and we do not generate any feasibility cut in this case. It is obvious that solving the MP with capacity constraints is more difficult than solving uncapacitated version. However, it enables us to obtain a decomposition algorithm where no feasibility cut is generated.

The master problem of BDTA can be given as:

$$\min \sum_{j\in I} f_j x_{jj} + \sum_{i\in I}\sum_{j\in I: i\neq j} d_{ij}x_{ij}(\sum_{m\in I}(\alpha t_{im} + \beta t_{mi})) + \theta$$

(MP) s.t.

$$(5.2), (5.3), (5.5)$$

$$\theta \geq \eta_{l_0} - \sum_{i\in I} \eta_{l_i} x_i \quad \forall l \in \omega_{opt}. \tag{5.21}$$

$$\theta \geq 0. \tag{5.22}$$

The definition of the auxilary variable θ and optimality cuts are the same as used in the decomposition algorithm for QCHLP.

To generate optimality cuts in this case, we use two options: *LL cuts* used in Integer L-shaped algorithm and problem specific optimality cuts (TBB cuts). In *LCQHLP*, we can only use the maximum amount of decrease on the current backbone traffic cost by changing the value of a variable from 1 to 0.

As the capacity constraints are linear in LCQHLP, we don't use the *traffic bound* algorithm in this case. Therefore, optimality cut is generated by only considering maximum reduction on θ_r value.

In BDTA for LCQHLP, we use the following inequality as optimality cuts:

$$\theta \geq \theta_r(\overline{x}) - \sum_{i \in I} \sum_{j \in I: \overline{x}_{ij}=1} \sum_{m \in I} \max_{l \in I}\{R_{jl}\}(t_{im} + t_{mi})(1 - x_{ij}) \qquad (5.23)$$

where $\theta_r(\overline{x})$ is the current backbone traffic cost calculated for a given master problem solution, $\overline{x}$. In the subproblem phase, we evaluate the current backbone traffic cost with a given master problem's solution, $\overline{x}$, and add optimality cuts when they are necessary (e.g, $\theta < \theta(\overline{x})$). The flowchart for the overall BDTA for $LCQHLP$ is given in Figure 5.2.

We give computational results of BDTA with TBB cuts and BDTA with LL cuts for LCQHLP in Tables 5.18 and 5.19. Average CPU time over 45 instances is 1,216.7 sec. and 4,912.1 sec for BDTA with TBB cuts and BDTA with LL cuts, respectively. In terms of average CPU time, using TBB cuts in the decomposition algorithm is superior to using LL cuts. Moreover, for each instance which is solved optimally, BDTA with TBB cuts requires less computational time than BDTA with LL cuts. Except the instances (Ins 31 and Ins 35) in Table 5.19, BDTA with TBB cuts finds a better solution than BDTA with LL cuts. For each instance, the number of optimality cuts used in BDTA with TBB cuts is less than the one in the method using LL cuts.

Figure 5.2: Flowchart of Branch-and-check for *LCQHLP*

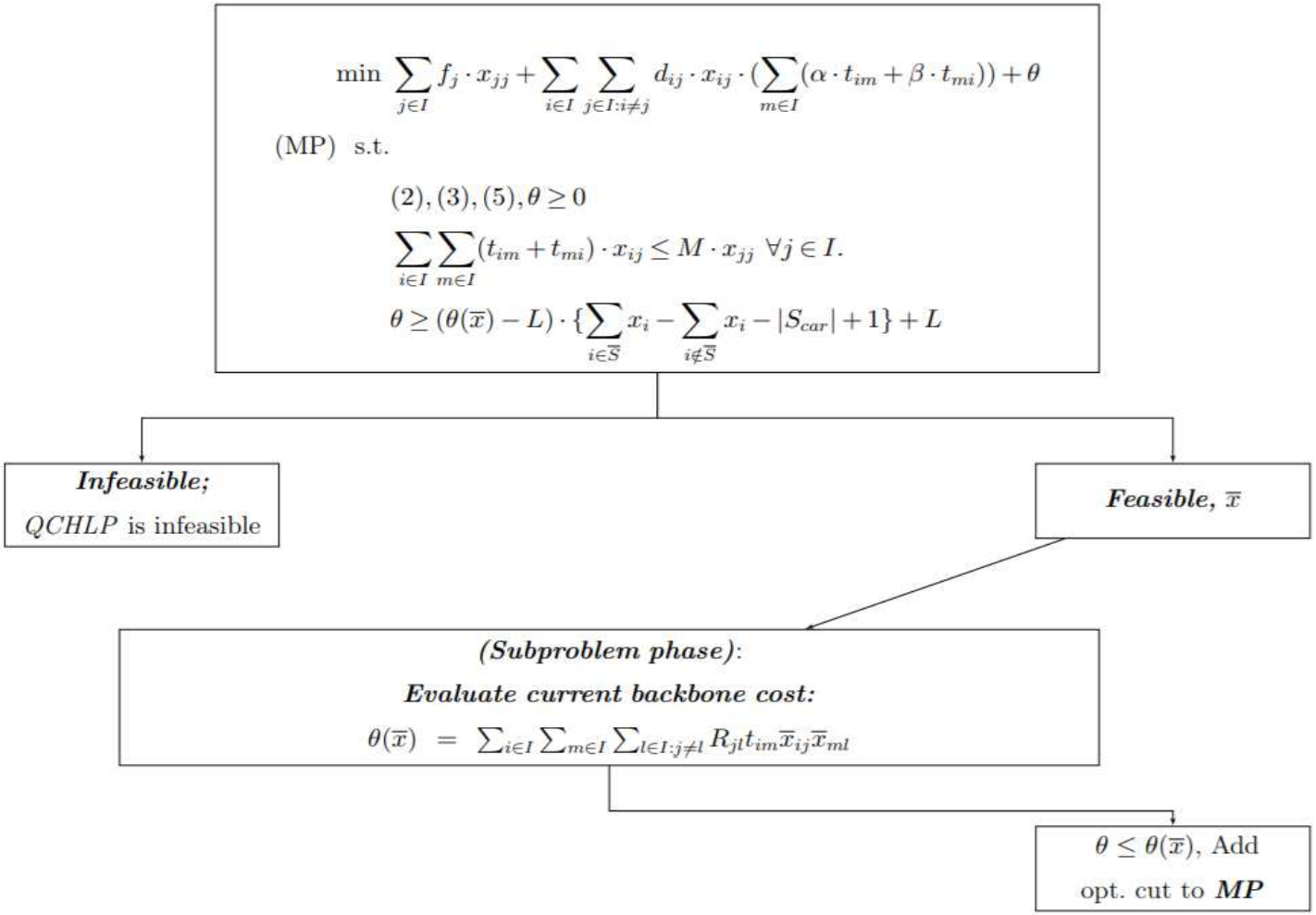


Table 5.18: Computational results of *LL* optimality cuts and TBB optimality cuts on *LCQHLP* for 20 nodes and 25 nodes

				BDTA with TBB cuts						*BDTA with LL cuts*						
Ins	Cap	Nodes	FC	Obj	CPU	B & B Nodes	R.Gap	Hubs	OptCut	Obj	CPU	Nodes	Sts	R.Gap	Hubs	OptCut
1	1500	20	L	160,671	**8.78**	11420	0	7;11;14	**208**	160,671	7200.55	208970	ATL	0.04	7;11;14	11876
2				99,689	**3**	1332	0	11;14	**123**	99,689	1555.11	28685	OptTol	0	11;14	5184
3				70,515	**0.02**	0	0	11	**0**	70,515	0.02	0	Opt	0	11	2
4	1500	20	T	486,423	**37.34**	36917	0	7;10;11,14;18	**535**	486,423	7200.45	217,212	ATL	0.04	7;10;11;14;18	12294
5				174,840	**2.53**	2109	0	7;11;18	**78**	174,840	7200.61	94,227	ATL	0.03	7;11;18	14366
6				122,807	**0.98**	425	0	11;18	**67**	122,807	179.02	8,783	OptTol	0	11;18	1899
7				104,814	**1.81**	488	0	10;11	**101**	104,814	219,75	8,948	Opt	0	10;11	1878
8				71,117	**0.02**	0	0	11	**2**	71,117	0.02	0	Opt	0	11	2
9	3270	20	T	266,627	**2.78**	2706	0	7;11;18	**175**	266,627	7200.53	133001	ATL	0.06	7;11;18	14048
10				157,911	**1.39**	508	0	11;18	**93**	157,911	456.68	18114	Opt	0	11;18	2946
11				103,093	**0.02**	0	0	10	**2**	103,093	0.03	0	Opt	0	10	2
12	3270	20	L	234,008	**7.43**	6775	0	6;11,14	**188**	234,008	7200.25	125219	ATL	0.05	6;11;14	15422
13				140,665	**0.44**	134	0	7;14	**21**	140,665	202.031	14965	Opt	0	7;14	1993
14				100,072	**0**	0	0	10	**2**	100,072	0.06	0	Opt	0	10	2
15	1500	25	L	289,152	**7200.1**	997834	0.06	3;8;14;16;18;24	**9492**	289,152	7200.59	122170	ATL	0.11	3;8;14;16;18,24	12619
16				161,361	**192.44**	65236	0	12;14;24	**912**	161,917	7200.81	125170	ATL	0.07	9;16,24	10457
17				91,430	**0.5**	109	0	8;25	**28**	91,430	7201.44	72942	ATL	0.03	8;25	9176
18				96,823	**3.39**	628	0	9,24	**49**	96,823	7200.89	49399	ATL	0.02	9;24	9086
19				70,785	**0.03**	0	0	13	**2**	70,785	0.04	0	0	0	13	2
20	1500	25	T	383982	**1633**	652885	0	9;11;13;14;24	**2071**	383,982	7200.42	396069	ATL	0.06	9;11;13;14;24	5613
21				208,008	**53.48**	27924	0	11;14;24	**202**	208,008	7200.77	52806	ATL	0.05	11;14;24	11786
22				94,393	**0.05**	0	0	25	**2**	94,393	0.03	0	Opt	0	25	2
23	3270	25	T	301,164	**12.33**	12454	0	9,11;24	**221**	301,164	7201	74514	ATL	0.07	9;11;24	12211
24				183,027	**3.25**	769	0	11;13	**88**	183,027	7200.6	74324	ATL	0.02	11;13	9084
25				94,393	**0.03**	0	0	25	**2**	94,393	0.02	0	Opt	0	25	2
26	3270	25	L	241,410	**214.44**	130597	0	8;14;18	**1497**	241,817	7200.73	116683	ATL	0.09	7;14;18	12607
27				143,779	**1.53**	370	0	9;18	**70**	143,779	7200.53	271390	ATL	0.04	9;18	6901
28				91,430	**0.56**	87	0	8,25	**29**	91,430	7200.81	70766	ATL	0.03	8;25	8605
29				82,528	**0.06**	0	0	13	**2**	82,528	0.016	0	Opt	0	13	2

Table 5.19: Computational results of *LL* optimality cuts and TBB optimality cuts on *LCQHLP* for 40 nodes

				BDTA with TBB cuts						*BDTA with LL cuts*						
Ins	Cap	Nodes	FC	Obj	CPU	B & B Nodes	R.Gap	Hubs	OptCut	Obj	CPU	Nodes	Sts	R.Gap	Hubs	OptCut
30	1500	40	L	296,896	**7206.2**	376900	0.11	6;11;22;25;28;29	**4395**	296,896	7,201	117251	ATL	0.12	6;11;22;25;28;29	4752
31				162,709	**7201.6**	383717	0.02	14;19;29	**7343**	162,383	7,200	170363	ATL	0.07	14;19;29	4186
32				113,418	**16.83**	1527	0	14,29	**94**	113,418	7,200	124399	ATL	0.04	14;29	3434
33				98,017	**38.09**	3907	0	14;29	**345**	98,017	7,202	48847	ATL	0.04	14;29	5476
34				87,480	**95.64**	36143	0	14;19	**480**	87,481	7201.33	53116	ATL	0.02	14;19	5024
35	1500	40	T	402,667	**7206.1**	2150401	0.09	14;19;22;35;38	**980**	402,443	7200.83	259851	ATL	0.11	19;22;35;38	2983
36				203,798	**7200.1**	412096	0.007	14;19;22	**7318**	203,798	7200.31	55913	ATL	0.03	14;19;22	7322
37				139,354	**67.67**	12823	0	14;19	**460**	139,354	7201.34	49101	ATL	2	14;19	5992
38				118,916	**369.56**	58980	0	14;22	**2119**	118,981	7200.77	34625	ATL	0.02	14;22	5123
39				106,025	**352.92**	74462	0	14;22	**1867**	106,025	7202.19	41453	ATL	0.01	14;22	5699
40	3270	40	T	312,435	**7200.1**	524951	0.006	14;19;22	**6390**	312,435	7202.03	52907	ATL	0.07	14,19;22	7123
41				174,975	**18.86**	2070	0	14;19	**199**	174,975	7203.53	50038	ATL	0.02	14;19	6358
42				118,645	**0.13**	0	0	19	**2**	118,465	**0.13**	0	Opt	0	19	2
43	3270	40	L	242,828	**7200.1**	984437	0.02	11;22;28	**3933**	242,828	7200.91	92526	ATL	0.09	11;22;28	6436
44				143,442	**15.23**	1545	0	14;29	**156**	143,442	7201.09	54867	ATL	0.05	14,29	5643
45				102,702	**0.17**	0	0	19	**2**	102,072	**0.08**	102702	Opt	0	19	2

5.4 Conclusion

In this chapter, we consider the exact solution methods for solving QCHLP which is a variant of single allocation hub location problem. This problem includes both nonlinear objective function and capacity constraints. Nonlinear terms in the objective function represents total backbone traffic cost, while nonlinear capacity constraints are due to the amount of backbone traffic. In the literature, the only exact solution method for QCHLP is a branch-and-cut method proposed by [53]. In their algorithm, they use a two-index variable but exponential number of constraints exist to be seperated in the algorithm. Different than this approach, we employed the idea behind logic based Benders decomposition in which generating cuts does not depend on the LP duality theory.

To the best of our knowledge, BDTA that we proposed for QCHLP is the first Benders decomposition implementation. We don't use any linearizations for the objective function and the capacity constraints. Therefore, this decomposition methods provide us to use only two-indexed variables in the model. By using the problem structure, we generate problem specific optimality cuts in addition to the feasibility cuts generated for QCHLP in Chapter 4. To increase the convergence speed of the algorithm, we implement this algorithm in a single branch-and-bound tree of the MP as done in branch-and-Benders cut or branch and check algorithms.

In this chapter, we demonstrate that BDTA outperforms the MIQCP solver of CPLEX for especially large size instances or difficult instances. By using BDTA, we could solve the instances that are not solved to optimum within reasonable CPU time in MIQCP. In the computational study, we observe that the performance of the problem specific cuts is better than the one of regular optimality cuts (LL cuts) in ILS algorithm for most of the instances. The reason is that BDTA with TBB cuts requires less number of optimality cuts on average than using LL cuts.

As the master problem includes both decision variables (*hub location and the assignment*), we could evaluate the amount of backbone traffic and backbone

traffic cost for a MP solution. The decomposition algorithm that we propose for QCHLP is based on this idea. Therefore, nonlinear integer problems which have similar structure with QCHLP, could be solved to optimum by using the idea behind BDTA.

CHAPTER 6

CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

In this chapter, firstly, we summarize the main contributions of each chapter of this dissertation in Section 6.1. Then, we give possible research directions in Section 6.2.

6.1 Main Contributions of the Dissertation

In this dissertation, we have studied three discrete location and hub location problems in which the objective function and/or the constraints include nonlinear terms. Although most of the studies in location theory consider linear problems, nonlinearity is a common issue that can be encountered in real life applications. The problems that we considered in Chapter 3 and 4 include nonlinear constraints. In the last problem given in Chapter 5, the mathematical model has both nonlinear objective function and nonlinear constraints. All problems are in the class of nonlinear integer programs which are known as very challenging problems. Therefore, solving these problems to optimality by using efficient algorithms is one of the main contributions of this dissertation. The common property for the proposed algorithms is that decomposition algorithms are based on Benders decomposition and cut generation procedure relies on the problems' structures.

In Chapter 3, we consider a nonliear wireless local area network design problem with two different linear objective functions. Due to technical restrictions

in the system, the mathematical model includes nonlinear capacity constraints. For both variants of the problem (WLANDP1 and WLANDP2), we proposed a MISOCP reformulation. This reformulation provides us to solve the model by using a commercial solver. One of the advantages of using this formulation is that we can use this reformulation regardless of the objective function. However, in the literature, the algorithms developed for solving this model depends on the objective function.

For WLANDP1, we also propose a Benders decomposition type algorithm. In the algorithm for WLANDP1, problem specific feasibility and general purpose optimality cuts are used. The algorithm is implemented in a branch-and-bound tree of the MP. Adding feasibility and optimality cuts in a single branch and bound tree of the MP increases convergence speed of the algorithm. However, MISOCP demonstrates a better performance than BDTA1 in our computational study.

For WLANDP2, we develop a branch-and-Benders decomposition approach in which feasibility cuts are generated in a more advanced procedure. As we expected, the relative performance of branch-and-Benders cut approach is better than that of BDTA2. As the subproblem doesn't have any objective function, only feasibility cuts were added to the MP in this case. From our computational study, we conclude that the performance of BDTAs becomes worse when more feasibility cuts are necessary.

Another nonlinear integer problem is considered in Chapter 4. The problem can arise in telecommunication network systems and called quadratic capacitated concentrator location problem. QCCLP includes a linear objective function and nonlinear capacity constraints. The same logic behind the decomposition algorithms applied for WLANDPs is also used for QCCLP.

In Chapter 4, without using any linearizations of nonlinear capacity constraints, we develop an exact Benders decomposition based algorithm. In the problem, MP is a relaxation of the original problem and the subproblem is a feasibility check. For the branch-and-check algorithm for QCCLP, we used several enhancement steps and evaluated the effects of these steps on the performance of

the algorithm. We generated valid inequalities by using the problem's structure. We observed that adding valid inequalities to the MP is one of the most important steps in terms of convergence speed of the algorithm.

One of the most important results that can be inferred from our computational study is that strong valid inequalities and feasibility cuts must be generated for difficult instances. We test the performance of branch-and-check algorithm with MIQCP solver of CPLEX. We demonstrate that we can solve large size instances to optimality by using a variant of branch-and-check alternative, while they couldn't be solved optimally by MIQCP solver.

The last problem in this dissertation is QCHLP which is a general version of QCCLP. We give our results for QCHLP in Chapter 5. The decomposition algorithm that we develop for QCHLP is an extension of the branch-and-check method for QCCLP. As optimality cuts, we develop problem specific optimality cuts. To evaluate the performance of this optimality cut, we used LL optimality cuts. In the computational study, there are instances in which BDTA with TBB cuts gives better solutions than BDTA with LL cuts. On the other hand, these two alternative cuts can have similar performance on some subset of instances. To observe if there exists a significant difference between two cuts, we test them on LCQHLP which is a special case of QCHLP.

From the computational results, we conclude that BDTA with TBB cuts is superior to the other alternative (BDTA with LL cuts) in terms of average CPU time. As we expected, BDTA with TBB cuts requires less number of optimality cuts.

To sum up, we develop the first MISOCP reformulation of the nonlinear constraints for WLANDP1 and WLANDP2. Both branch-and-check algorithm and BDTAs are the first Benders decomposition based algorithms for QCCLP and QCHLP, respectively. When compared the relative performance of BDTAs with that of automatic Benders decomposition, we observed that adding problem specific cuts demonstrates a better performance than automatic implementation for QCCLP.

6.2 Future Research Directions

In this dissertation, we studied nonlinear integer location and hub location problems in which all parameters are deterministic. Although the problems are in the class of deterministic optimization problems, we used the logic behind the decomposition algorithms which are widely used for two stage stochastic programs. Adding stochasticity to nonlinear integer problems will increase the complexity of the problems. Therefore, solving such kind of problems to optimum will be more challenging within reasonable time limit.

For both variants of the problem studied in Chapter 3, we developed BDTAs. As a future research, demand of the user terminals could be generated from a finite set of alternatives. In this case, power level selection decisions can be determined in the first stage. After uncertain demand values are realized, determining the assignments of user terminals to access points will be decision variables in the second stage. Resulting extensive model will be in the class of nonlinear stochastic integer programs. Proposed BDTAs can be used to solve this problem to optimum. Because, in Chapter 3, we develop the algorithm by assuming that we have only one scenario for each parameter. However, when the number of scenarios increases, it is neccessary to find efficient algorithms to solve integer subproblems. On the other hand, MISOCP reformulation that we develop for deterministic problem is also valid for stochastic variant of WLANDPs. In this case, the resulting model will include high number of decision variables and constraints. To overcome this drawback, decomposition algorithms in which subproblems can include conic constraints could be used.

In WLANDPs studied in this dissertation, the objective functions are linear. As a future research, nonlinear objective functions could be taken into consideration. Including nonlinear objective function to the model makes the problem more challenging to solve. Efficient decomposition algorithms or reformulations, which can be applied for the objective function could be implemented.

We studied QCCLP and developed an exact solution method based on Benders decomposition for this problem in Chapter 4. We assume a star network

between the terminal nodes and hub (concentrator) nodes is complete. As a future research, different network structures could be studied. Different structures have been widely studied in the hub location literature. When the structure of the network changes, objective function and constraints may include nonlinear terms. They can be handled by using the similar logic behind the proposed decomposition algorithms for QCCLP.

For QCCLP, we assume that all demand values are known. It is possible that there are finite set of scenarios from which demand values are generated. In this case, we could study a two stage stochastic integer program for the resulting problem. In the literature, different type of uncertainities (e.g., polyhedral uncertainty, hose demand uncertainty) are defined for the problems arising in telecommunication networks. When this type of uncertainities are added to the model, we may end up with nonlinear models. As an alternative for a future research, developing exact decomposition algorithms without using any linearizations can be studied. One other alternative study in the future could be considering the p-median variant of this problem.

In the hub location literature, there are similar problems with QCCLP. Modular link capacitated hub location model also has nonlinear terms in the capacity constraints. As in QCCLP, proposed nonlinear terms exists due to the multiplications of binary variables. The idea behind generating problem specific feasibility and optimality cuts could be used in the decomposition algorithms for modular link capacitated problem.

The last problem we studied in this dissertation is QCHLP which includes both nonlinear objective function and the capacity constraints. As an alternative for a future research, different types of network structures, different nonlinear terms and stochastic variants of the problem could be studied.

REFERENCES

[1] AP data set. http://users.monash.edu/~andrease/Downloads.htm. [Online; accessed 26-February-2018].

[2] M. Aktürk, A. Atamtürk, and S. Gürel. A strong conic quadratic reformulation for machine-job assignment with controllable processing times. *Operations Research Letters*, 37(3):187 – 191, 2009.

[3] F. Alizadeh and D. Goldfarb. Second-order cone programming. *Mathematical Programming*, 95:3–51, 2001.

[4] F. T. Altekin. A comparison of piecewise linear programming formulations for stochastic disassembly line balancing. *International Journal of Production Research*, 55(24):7412–7434, 2017.

[5] S. Alumur and B. Y. Kara. Network hub location problems: The state of the art. *European Journal of Operational Research*, 190(1):1 – 21, 2008.

[6] G. Angulo, S. Ahmed, and S. S. Dey. Improving the integer l-shaped method. *Informs Journal on Computing*, 28(3):483 – 499, 2016.

[7] A. Atamtürk, G. Berenguer, and Z.-J. M. Shen. A conic integer programming approach to stochastic joint location-inventory problems. *Operations Research*, 60(2):366–381, 2012.

[8] T. Aykin. Lagrangian relaxation based approaches to capacitated hub-and-spoke network design problem. *European Journal of Operational Research*, 79(3):501 – 523, 1994.

[9] T. Aykin. Networking policies for hub-and-spoke systems with application to the air transportation system. *Transportation Science*, 29(3):201–221, 1995.

[10] J. C. Beck. Checking-up on branch-and-check. 2010.

[11] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numer. Math.*, 4(1):238–252, 1962.

[12] H. Y. Benson and Ümit Sağlam. *Mixed-Integer Second-Order Cone Programming: A Survey*, chapter Chapter 2, pages 13–36.

[13] A. Bockmayr and N. Pisaruk. Detecting infeasibility and generating cuts for mixed integer programming using constraint programming. *Computers and Operations Research*, 33(10):2777 – 2786, 2006.

[14] P. Bonami, M. Kılınç, and J. Linderoth. Algorithms and software for convex mixed integer nonlinear programs. In J. Lee and S. Leyffer, editors, *Mixed Integer Nonlinear Programming*, volume 154 of *The IMA Volumes in Mathematics and its Applications*, pages 1–39. Springer New York, 2012.

[15] D. L. Bryan and M. E. O'Kelly. Hub-and-spoke networks in air transportation: An analytical review. *Journal of Regional Science*, 39(2):275–295, 1999.

[16] R. S. d. Camargo and G. d. Miranda Jr. Addressing congestion on single allocation hub-and-spoke networks. *Pesquisa Operacional*, pages 465 – 496, 2012.

[17] J. Campbell. A survey of network hub location. 6:31–49, 01 1994.

[18] J. F. Campbell. Integer programming formulations of discrete hub location problems. *European Journal of Operational Research*, 72(2):387 – 405, 1994.

[19] J. F. Campbell and M. E. O'Kelly. Twenty-five years of hub location research. *Transportation Science*, 46(2):153–169, 2012.

[20] G. Codato and M. Fischetti. Combinatorial benders' cuts for mixed-integer linear programming. *Operations Research*, 54(4):756–766, 2006.

[21] I. Contreras. *Hub Location Problems*, pages 311–344. Springer International Publishing, 2015.

[22] I. Contreras, J.-F. Cordeau, and G. Laporte. Benders decomposition for large-scale uncapacitated hub location. *Operations Research*, 59(6):1477–1490, 2011.

[23] I. Contreras, J.-F. Cordeau, and G. Laporte. Exact solution of large-scale hub location problems with multiple capacity levels. *Transportation Science*, 46(4):439–459, 2012.

[24] I. Contreras, J. A. Díaz, and E. Fernández. Branch and price for large-scale capacitated hub location problems with single assignment. *INFORMS Journal on Computing*, 23(1):41–55, 2011.

[25] I. Contreras, J. A. Díaz, and E. Fernández. Lagrangean relaxation for the capacitated hub location problem with single assignment. *OR Spectrum*, 2009.

[26] Á. Corberán, J. Peiró, V. Campos, F. Glover, and R. Martí. Strategic oscillation for the capacitated hub location problem with modular links. *Journal of Heuristics*, 22(2):221–244, 2016.

[27] J.-F. Cordeau, F. Pasin, and M. M. Solomon. An integrated model for logistics network design. *Annals of Operations Research*, 144(1), 2006.

[28] A. I. Corréa, A. Langevin, and L.-M. Rousseau. Scheduling and routing of automated guided vehicles: A hybrid approach. *Computers and Operations Research*, 34(6):1688 – 1707, 2007.

[29] I. Correia, S. Nickel, and F. S. da Gama. Single-assignment hub location problems with multiple capacity levels. *Transportation Research Part B: Methodological*, 44(8):1047 – 1066, 2010.

[30] M. da Graça Costa, M. E. Captivo, and J. Clímaco. Capacitated single allocation hub location problem—a bi-criteria approach. *Computers and Operations Research*, 35(11):3671 – 3695, 2008.

[31] R. de Camargo and G. Miranda. Single allocation hub location problem under congestion: Network owner and user perspectives. *Expert Systems with Applications*, 39(3):3385 – 3391, 2012.

[32] R. de Camargo, G. Miranda, and H. Luna. Benders decomposition for the uncapacitated multiple allocation hub location problem. *Computers and Operations Research*, 35(4):1047 – 1064, 2008.

[33] R. de Camargo, G. Miranda, and H. Luna. Benders decomposition for the uncapacitated multiple allocation hub location problem. *Computers & Operations Research*, 35(4):1047 – 1064, 2008.

[34] R. S. de Camargo, G. de Miranda, and H. P. L. Luna. Benders decomposition for hub location problems with economies of scale. *Transportation Science*, 43(1):86–97, 2009.

[35] H. Şen and K. Bülbül. A strong preemptive relaxation for weighted tardiness and earliness/tardiness problems on unrelated parallel machines. *INFORMS Journal on Computing*, 27(1):135–150, 2015.

[36] A. Ernst and M. Krishnamoorthy. Solution algorithms for the capacitated single allocation hub location problem. *Annals of Operations Research*, pages 141–159, 1999.

[37] A. T. Ernst and M. Krishnamoorthy. Efficient algorithms for the uncapacitated single allocation p-hub median problem. *Location Science*, 4(3):139 – 154, 1996.

[38] R. Z. Farahani, M. Hekmatfar, A. B. Arabani, and E. Nikbakhsh. Hub location problems: A review of models, classification, solution techniques, and applications. *Computers and Industrial Engineering*, 64(4):1096 – 1109, 2013.

[39] M. M. Fazel-Zarandi and J. C. Beck. Solving a location-allocation problem with logic-based benders' decomposition. In *CP*, 2009.

[40] B. Gendron, R. Garroppo, G. Nencioni, M. Scutellà, and L. Tavanti. Benders decomposition for a location-design problem in green wireless local area networks. *Electronic Notes in Discrete Mathematics*, 41:367 – 374, 2013.

[41] B. Gendron, R. Garroppo, G. Nencioni, M. Scutellà, and L. Tavanti. A branch and benders-cut method for nonlinear power design in green wireless local area networks. *CIRRELT, Technical Report*, 2014.

[42] B. Gendron, M. G. Scutellà, R. G. Garroppo, G. Nencioni, and L. Tavanti. A branch-and-benders-cut method for nonlinear power design in green wireless local area networks. *European Journal of Operational Research*, 255(1):151 – 162, 2016.

[43] S. Gürel. A conic quadratic formulation for a class of convex congestion functions in network flow problems. *European Journal of Operational Research*, 211(2):252 – 262, 2011.

[44] G. Heilporn, J.-F. Cordeau, and G. Laporte. An integer l-shaped algorithm for the dial-a-ride problem with stochastic customer delays. *Discrete Applied Mathematics*, 159(9):883 – 895, 2011.

[45] J. Hooker and G. Ottosson. Logic-based benders decomposition. *Mathematical Programming*, 96:33–60, 2003.

[46] J. N. Hooker. Planning and scheduling by logic-based benders decomposition. *Operations Research*, 55(3):588–602, 2007.

[47] V. Jain and I. E. Grossmann. Algorithms for hybrid milp/cp models for a class of optimization problems. *INFORMS Journal on Computing*, 13(4):258–276, 2001.

[48] J. G. Klincewicz. Avoiding local optima in thep-hub location problem using tabu search and grasp. *Annals of Operations Research*, 40(1):283–302, 1992.

[49] J. G. Klincewicz. Hub location in backbone/tributary network design: a review. 6:307 – 335, 1998.

[50] E. Koca, H. Yaman, and M. S. Aktürk. Stochastic lot sizing problem with controllable processing times. *Omega*, 53:1 – 10, 2015.

[51] J. Kratica, M. Milanović, Z. Stanimirović, and D. Tošić. An evolutionary-based approach for solving a capacitated hub location problem. *Applied Soft Computing*, 11(2):1858 – 1866, 2011.

[52] Y.-J. Kuo and H. D. Mittelmann. Interior point methods for second-order cone programming and or applications. *Computational Optimization and Applications*, 28(3):255–285, 2004.

[53] M. Labbé, H. Yaman, and E. Gourdin. A branch and cut algorithm for hub location problems with single assignment. *Mathematical Programming*, 102(2):371–405, Mar 2005.

[54] G. Laporte and F. V. Louveaux. The integer l-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13(3):133 – 142, 1993.

[55] G. Laporte, F. V. Louveaux, and L. van Hamme. An integer l -shaped algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research*, 50(3):415–423, 2002.

[56] G. Laporte, F. V. Louveaux, and L. van Hamme. An integer l-shaped algorithm for the capacitated vehicle routing problem with stochastic demands. *Operations Research*, 50(3):415–423, 2002.

[57] J. F. Meier and U. Clausen. Solving single allocation hub location problems on euclidean data. *Transportation Science*, 0(0):null, 0.

[58] M. Meraklı and H. Yaman. Robust intermodal hub location under polyhedral demand uncertainty. *Transportation Research Part B: Methodological*, 86:66 – 85, 2016.

[59] M. Meraklı and H. Yaman. A capacitated hub location problem under hose demand uncertainty. *Computers and Operations Research*, 88:58 – 70, 2017.

[60] N. Noyan, B. Balcik, and S. Atakan. A stochastic optimization model for designing last mile relief networks. *Transportation Science*, 50(3):1092–1113, 2016.

[61] M. O'Kelly, D. Skorin-Kapov, and J. Skorin-Kapov. Lower bounds for the hub location problem. *Management Science*, 41(4):713–721, 1995.

[62] M. E. O'Kelly. A quadratic integer program for the location of interacting hub facilities. *European Journal of Operational Research*, 32(3):393 – 404, 1987.

[63] M. E. O'Kelly and H. J. Miller. The hub network design problem: A review and synthesis. *Journal of Transport Geography*, 2(1):31 – 40, 1994.

[64] J. Penuel, J. C. Smith, and Y. Yuan. An integer decomposition algorithm for solving a two-stage facility location problem with second-stage activation costs. *Naval Research Logistics (NRL)*, 57(5):391–402, 2010.

[65] R. Rahmaniani, T. G. Crainic, M. Gendreau, and W. Rei. The benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259(3):801 – 817, 2017.

[66] I. Rodríguez-Martín and J. J. Salazar-González. Solving a capacitated hub location problem. *European Journal of Operational Research*, 184(2):468 – 479, 2008.

[67] B. Rostami, C. Buchheim, J. F. Meier, and U. Clausen. Lower bounding procedures for the single allocation hub location problem. *Electronic Notes in Discrete Mathematics*, 52:69 – 76, 2016.

[68] B. Rostami, N. Kämmerling, C. Buchheim, and U. Clausen. Reliable single allocation hub location problem under hub breakdowns. *Computers and Operations Research*, 96:15 – 29, 2018.

[69] P. Rubin. Benders Decomposition Then and Now. `https://orinanobworld.blogspot.com.tr/2011/10/benders-decomposition-then-and-now.html`, 2011. [Online; accessed 26-February-2018].

[70] R. Sadykov and L. A. Wolsey. Integer programming and constraint programming in solving a multimachine assignment scheduling problem with deadlines and release dates. *INFORMS Journal on Computing*, 18(2):209–217, 2006.

[71] S. Sen. Algorithms for stochastic mixed-integer programming models. In K. Aardal, G. Nemhauser, and R. Weismantel, editors, *Discrete Optimization*, volume 12 of *Handbooks in Operations Research and Management Science*, pages 515 – 558. Elsevier, 2005.

[72] S. Sen and L. J. Higle. The c3 theorem and a d2 algorithm for large scale stochastic mixed-integer programming: Set convexification. *Mathematical Programming*, 104(1):1–20, 2005.

[73] H. D. Sherali and J. C. Smith. An improved linearization strategy for zero-one quadratic programming problems. *Optimization Letters*, 1(1):33–47, 2007.

[74] D. Skorin-Kapov and J. Skorin-Kapov. On tabu search for the location of interacting hub facilities. *European Journal of Operational Research*, 73(3):502 – 509, 1994.

[75] D. Skorin-Kapov, J. Skorin-Kapov, and M. O'Kelly. Tight linear programming relaxations of uncapacitated p-hub median problems. *European Journal of Operational Research*, 94(3):582 – 593, 1996.

[76] E. S. Thorsteinsson. Branch-and-check: A hybrid framework integrating mixed integer programming and constraint logic programming. 2001.

[77] D. Wheatley, F. Gzara, and E. Jewkes. Logic-based benders decomposition for an inventory-location problem with service constraints. *Omega*, 55:10 – 23, 2015.

[78] H. Yaman. Concentrator location in telecommunications networks. *Combinatorial Optimization*, 2005.

[79] H. Yaman and G. Carello. Solving the hub location problem with modular link capacities. *Computers and Operations Research*, 32(12):3227 – 3245, 2005.

APPENDIX A

MIQCP IN CPLEX 12.7.1

Table A.1: Mathematical Model Types with different objective and constraints

Obj /Const	Linear const.	Nonlinear const.
Linear obj	*MILP*	***MIQCP***
Nonlinear obj	*MIQP*	***MIQCP***

Table A.1 gives the suitable method depending on type of the objective function and constraints in any model.

As we compare the performance of MIQCP solver and decomposition algorithms, we give a brief background on MIQCPs in this part. IBM CPLEX can solve MIQCPs, if constraints and objective function have certain conditions. MIQCP with a linear objective function is solved by CPLEX MIQCP solver. When the objective function of the model includes quadratic terms and the terms are convex, CPLEX solves the model to optimum. On the other hand, if the objective function includes a multiplication of binary variables, it is also solved through CPLEX. This situation implies that the objective function could be a non-convex function.

There are also *two conditions* on the constraints of MIQCP.

- If the nonlinear constraints can be reformulated as second order cone programs, MIQCP solves such kind of models to optimum.
- The nonlinear constraints could be non-convex. However, in this case, the only condition is that nonlinear terms include multiplications of binary variables.

As the mathematical model of QCCLP considered in this paper has a linear objective function and nonlinear constraints, the suitable solver for our problem is MIQCP.

Through solving MIQCPs in CPLEX, there is a parameter in which the strategy of solving MIQCPs is selected. If the user does not determine the related parameter, CPLEX will choose the strategy automatically. There are other two options of this strategy. **QCP relaxation** is solved at each node of the model, when the parameter is set to value of 1. If the parameter 2 is selected, **LP relaxation** is solved at each node instead of QCP relaxation. The performance of selected strategy depends on the problem's structure under consideration. To determine the best strategy to solve MIQCPs, two possible options should be examined on the model.

In this study, we chose the parameter 2 in which LP relaxation is solved at each node, since the performance of parameter 2 is better than the performance of parameter 1 for all instances in computational study.

In this study, we consider a capacitated concentrator location problem in which the objective function is linear and the constraints defining capacity restrictions between non-hub nodes and hubs are nonlinear. The novelty of the problem comes from capacity usage of hubs. The capacity of the hubs is also used for backbone traffic that occurs between hub locations.

- Linearization of quadratic terms on the constraints

  By using standard linearization techniques for multiplications of binary variables, we give two linearized models to solve the problem quadratic capacitated concentrator location problem (QCCLP) in an extensive form. One of the linearization techniques was developed by (Dantzig, 1959) and the other one was given in the study of [75]. In the literature, the quadratic term can be seen in various areas such as telecommunication problems, quadratic assignment and quadratic knapsack problem.

  Lets first give a linearization for the objective function's last term including multiplication of binary variables.

By defining $z_{jl} \geq \sum_{i \in I} \sum_{m \in I} t_{im} \cdot x_{ij} \cdot x_{ml}, \ \ \forall j, l \in I$. We can write the last term on the objective function as $\sum_{j \in I} \sum_{l \in I: j \neq l} R_{jl} \cdot z_{jl}$, where z_{jl} define the total traffic on backbone link from j to l. Lets define $x_{ijml} = x_{ij} \cdot x_{ml}$.

$$x_{ijml} \geq x_{ij} + x_{ml} - 1, \ \ \forall i, j, m, l \in I.$$

$$x_{ijml} \leq x_{ij}, \ \ \forall i, j, m, l \in I.$$

$$x_{ijml} \leq x_{ml}, \ \ \forall i, j, m, l \in I.$$

$$z_{jl} \geq \sum_{i \in I} \sum_{m \in I} t_{im} \cdot x_{ijml}, \ \ \forall j, l \in I.$$

By using the constraints defined above, we need to write O(n^4) constraints.

Using variable z_{jl}, we can linearize the capacity constraints. The steps traced for linearization are given in [79].

As mentioned in ([73]) for general zero one quadratic program and ([79]) for modular link capacitated hub location problem, constraints ($x_{ijml} \leq x_{ij}$) and ($x_{ijml} \leq x_{ml}$) are redundant due to the fact that R_{jl} values are nonnegative.

For each j,

$$\sum_{l \in I: l \neq j} z_{jl} = \sum_{l \in I: l \neq j} \sum_{i \in I} \sum_{m \in I} t_{im} x_{ij} x_{ml} = \sum_{i \in I} \sum_{m \in I} t_{im} \cdot x_{ij} \sum_{l \in I: l \neq j} x_{ml}$$

.

Since $\sum_{l \in I: l \neq j} x_{ml} + x_{mj} = 1$ from constraints (5.2), we have $\sum_{l \in I: l \neq j} z_{jl} = \sum_{i \in I} \sum_{m \in I} t_{im} \cdot x_{ij} \cdot (1 - x_{mj})$.

Similarly, z_{lj} can be rewritten as,

$$\sum_{l \in I: l \neq j} z_{lj} = \sum_{l \in I: l \neq j} \sum_{i \in I} \sum_{m \in I} t_{mi} \cdot x_{ij} \cdot x_{ml} = \sum_{i \in I} \sum_{m \in I} t_{mi} \cdot x_{ij} \sum_{l \in I: l \neq j} x_{ml}$$

$$\sum_{l \in I: l \neq j} (z_{jl} + z_{lj}) = \sum_{i \in I} \sum_{m \in I} \{t_{im} \cdot x_{ij} \cdot (1 - x_{mj}) + t_{mi} \cdot x_{ij} \cdot (1 - x_{mj})\} \tag{A.1}$$

By using (A.1), we can rewrite Constraints (5.4) in a linear form.

$$\sum_{i\in I}\sum_{m\in I}(t_{im}+t_{mi})\cdot x_{ij}+\sum_{l\in I:l\neq j}(z_{jl}+z_{lj})\leq Q_h(j)\cdot x_{jj},\quad \forall j\in I. \quad \text{(A.2)}$$

After linearizing the constraints (5.4), we obtain the following linearized model which is called *Linearization 1 (LSAHLP-1)* in this paper.

This linearized formulation has $(n^4+2\cdot n^2)$ variables of which (n^4+n^2) are binary and it includes $(3\cdot n^4+2\cdot n^2+2\cdot n)$ linear constraints. Since all assignment and hub location decisions are binary, $(x_{ij}\in\{0,1\})$, there is no need to define x_{ijml} as binary variables. Therefore, we can use continuous variables $(0\leq x_{ijml}\leq 1)$ for multiplications of binary variables.

In the formulation *LSAHLP-2*, we can use the following constraints and obtain another *MILP* model for the problem *QSAHLP*.

$$\sum_{l\in I}x_{ijml}=x_{ij},\ \ \forall i,j,m\in I. \quad \text{(A.3)}$$

$$\sum_{j\in I}x_{ijml}=x_{ml},\ \ \forall i,l,m\in I. \quad \text{(A.4)}$$

Constraints (A.3) ensure that if a non-hub node i is assigned to a hub node j, (e.g. $x_{ij}=1$), the demand from node i to m should pass through another hub l to which node m is assigned. Similarly, if x_{ml} is 1, the flow between i and m should pass through the hub j. This is satisfied by constraints (A.4).

When the number of nodes in the network increases, the resulting linearized models (*LSAHLP-1 & LSAHLP-2*) are large scale mixed integer models. Therefore, they cannot be solved to optimum within reasonable CPU time, even if we have moderate size problems. Decomposition algorithms could be used in order to eliminate this difficulty. The main idea behind using decomposition structure is to handle nonlinearities in the capacity constraints without defining any new variable as done in linearizations.

For each instance, we report the following items in Tables A.2 and A.3:

- *Objective function value*: If the instance is solved to optimality, objective function gives the optimal solution's value.
- *CPU time for MIQCP*: As we do not use any specific cuts through solving the model in an extensive form, CPU time gives the overall computational time.
- *R.gap*: For each method, R.Gap is calculated the relative difference between the objective function value of the instance when the algorithm stops and best integer solution found.
- *T(i)*: This column gives the total computational time to find traffic bound values for backbone traffic under the assumption of being node i is a hub.
- *T(i)(j)*: Total required time spent to find minimum amount of backbone flow between any pair i and j in the network.
- *B1(cpu)*: This column gives total CPU time of $BV_2T_2F_1$ algorithm by adding traffic bound calculations' time. In other words, B1(cpu)= cpu(under B1 method)+ $T(i)$+ $T(i)(j)$. Similarly, B8(cpu)= cpu(under B8 method)+ $T(i)$.
- *Fcuts*: As the Benders decomposition algorithm includes feasibility cuts, these columns give total number of feasibility cuts added to the MP during the algorithm.
- *Best Method*: This column gives the method whose CPU time is the best over these methods. When an instance is not solved to optimum, we report the method in which a better bound (objective function value) is found.

Table A.2: Comparison of MIQCP, $BV_2T_{IP}F_{MF}$ and $BV_1T_{IP}F_{MF}$ Results for 20 & 25 Nodes

					MIQCP				$BV_2T_{IP}F_{MF}$					TB (IP)			$BV_1T_{IP}F_{MF}$						
Ins	Demand	Cap	FC	Nodes	Obj	CPU	R.Gap	Hubs	Obj	CPU	R.Gap	Hubs	F.cut	T(i)	T(ij)	Tot (CPU)	Obj	CPU	R.Gap	Hubs	F.cut	Tot(CPU)	Best Method
1	D			20	243,550	10.19	0	3,9;11;14;19	243,550	0.94	0	3;9;11;14;19	66	0.47	47.22	48.63	243,550	1.38	0	3;9;11;14;19	451	1.85	$BV_1T_{IP}F_{EF}$
2	D/1.2	3000	L	20	211,700	14.58	0	3;11;13;14	211,700	2.63	0	3;11;13;14	218	0.63	52.53	55.79	211,700	2.08	0	3;11;13;14	694	2.71	$BV_1T_{IP}F_{EF}$
3	D/1.3			20	201,947	35.45	0	3;9;11;14	201,947	13.89	0	3;9;11;14	662	2.28	63.64	79.81	201,947	5.08	0	3;9;11;14	1341	7.36	$BV_1T_{IP}F_{EF}$
4	D/1.4			20	184,475	5.86	0	6;11;14	184,475	3.19	0	6;11;14	102	0.56	69.23	72.98	184,475	1.88	0	6;11;14	466	2.44	$BV_1T_{IP}F_{EF}$
5	D/1.5			20	173,226	8.74	0	3;11;14	173,226	0.88	0	3;11;14	45	0.86	66.61	68.35	173,226	3.67	0	3;11;14	1062	4.53	$BV_1T_{IP}F_{EF}$
6	D			20	292,451	14.06	0	1;8;10;11;18	292,451	0.31	0	1;8;10;11;18	17	0.47	47.22	48.00	292,451	3.08	0	1;8;10;11;18	1093	3.55	$BV_1T_{IP}F_{EF}$
7	D/1.2	3000	T	20	236,912	12.98	0	7;9;11;18	236,912	1.23	0	7;9;11,18	17	0.63	52.53	54.39	236,912	5.69	0	7;9;11;18	1579	6.32	$BV_1T_{IP}F_{EF}$
8	D/1.3			20	221,316	20.83	0	1;7;11;18	221,316	26.71	0	1;7;11;18	1431	2.28	63.64	92.63	221,316	9.41	0	1;7;11;18	1941	11.69	$BV_1T_{IP}F_{EF}$
9	D/1.4			20	198,834	3.8	0	7;11;18	198,834	0.28	0	7;11;18	1	0.56	69.23	70.07	198,834	0.64	0	7;11;18	119	1.20	$BV_1T_{IP}F_{EF}$
10	D/1.5			20	188,071	4.14	0	7;11;18	188,071	0.66	0	7;11;18	7	0.86	66.61	68.13	188,071	1.27	0	7;11;18	289	2.13	$BV_1T_{IP}F_{EF}$
11	D			20	223,990	3.89	0	3;9;11;14	223,990	0.8	0	3;9;11,14	61	0.64	68.05	69.49	223,990	0.63	0	3;9;11;14	172	1.27	$BV_1T_{IP}F_{EF}$
12	D/1.2	4000	L	20	193,755	5.08	0	6;11;14	193,755	0.92	0	6;11;14	53	0.97	65.64	67.53	193,755	0.7	0	6;11;14	216	1.67	$BV_1T_{IP}F_{EF}$
13	D/1.3			20	181,317	4.69	0	3;11;14	181,317	2.28	0	3;11;14	99	0.50	60.13	62.91	181,317	0.41	0	3;11;14	112	0.91	$BV_1T_{IP}F_{EF}$
14	D/1.4			20	162,588	0.08	0	7;14	162,588	0.09	0	7;14	1	0.27	64.31	64.67	162,588	0.02	0	7;14	1	0.29	MIQCP
15	D/1.5			20	155,237	0.13	0	7;14	155,237	0.05	0	7;14	0	0.48	56.25	56.78	155,237	0.02	0	7;14	0	0.50	MIQCP
16	D			20	248,725	4.56	0	7;11;18	248,725	0.44	0	7;11;18	18	0.64	68.05	69.13	248,725	0.89	0	7;11;18	247	1.53	$BV_1T_{IP}F_{EF}$
17	D/1.2	4000	T	20	214,325	1.2	0	7;11;18	214,325	0.34	0	7;11;18	8	0.97	65.64	66.95	214,325	0.38	0	7;11;18	112	1.35	MIQCP
18	D/1.3			20	201,150	3.28	0	1;11;18	201,150	1.89	0	1;11;18	77	0.50	60.13	62.52	201,150	0.16	0	1;11;18	34	0.66	$BV_1T_{IP}F_{EF}$
19	D/1.4			20	184,240	0.06	0	7;18	184,240	0.09	0	7;18	0	0.27	64.31	64.67	184,240	0.02	0	7;18	0	0.29	MIQCP
20	D/1.5			20	175,956	0.09	0	7;18	175,956	0.05	0	7;18	0	0.48	56.25	56.78	175,956	0.03	0	7;18	0	0.51	MIQCP
21	D			25	242,304	22.32	0	3;11;14;18;24	242,304	1.19	0	3;11;14;18;24	45	1.38	97.17	99.74	242,304	3.58	0	3;11;14;18;24	725	4.96	$BV_1T_{IP}F_{EF}$
22	D/1.2	3000	L	25	204,369	17.02	0	3;14;16;24	204,369	1.2	0	3;14;16;24	16	1.42	141.23	143.85	204,369	2.97	0	3;14;16;24	671	4.39	$BV_1T_{IP}F_{EF}$
23	D/1.3			25	195,595	13.5	0	3;14;16;24	195,595	3.73	0	3;14;16;24	72	0.88	122.47	127.08	195,595	4.59	0	3;14,16;24	594	5.47	$BV_1T_{IP}F_{EF}$
24	D/1.4			25	180,858	11.5	0	9;11;24	180,858	0.78	0	9;11;24	15	0.7	113.14	114.62	180,858	5.2	0	9;11;24	1058	5.9	$BV_1T_{IP}F_{EF}$
25	D/1.5			25	130,011	0.17	0	8;24	130,111	0.09	0	8;24	0	0.86	99.4	100.3	130,011	0.015	0	8;24	0	0.88	MIQCP
26	D			25	338,719	618.63	0	3;9;11;14;24	338,719	37.11	0	3;9;11;14;24	222	1.38	97.17	135.66	338,719	1133.33	0	3;9;11;14;24	20804	1134.71	$BV_2T_{IP}F_{EF}$
27	D/1.2	3000	T	25	273,172	45.36	0	9;11;14;24	273,172	1.47	0	9;11;14;24	12	1.42	141.23	144.12	273,172	101.48	0	9;11;14;24	4693	102.90	MIQCP
28	D/1.3			25	264,053	7200.17	0,0067	9;11;14;24	263,548	540.33	0	9;11;14;24	9584	0.88	122.47	663.68	264,053	7200.06	0.83	9;11;14;24	58994	7200.94	$BV_2T_{IP}F_{EF}$
29	D/1.4			25	229,511	9,25	0	11;14;24	229,511	0,72	0	11;14;24	3	0.7	113.14	114.56	229,511	34.45	0	11;14;24	1148	35.15	MIQCP
30	D/1.5			25	220,571	22.25	0	9;11;24	220,571	0.63	0	9;11;24	5	0,86	99,4	100.89	220,571	20,86	0	9;11;24	3445	21,72	$BV_1T_{IP}F_{EF}$
31	D			25	222,479	5.05	0	3;9;16;18	222,479	0.59	0	3;9;16;18	11	0,28	102,81	103.68	222,479	0.64	0	3;9;16;18	81	0.92	$BV_1T_{IP}F_{EF}$
32	D/1.2	4000	L	25	191,488	4.34	0	9;11;18	191,488	0.55	0	9;11;18	16	0.64	111.41	112.6	191,488	0.64	0	9;11;18	106	1.28	$BV_1T_{IP}F_{EF}$
33	D/1.3			25	181,654	7.44	0	9;11;18	181,654	4.13	0	9;11;18	101	0.8	120	124.93	181,654	0.81	0	9;11;18	158	0.81	$BV_1T_{IP}F_{EF}$
34	D/1.4			25	168,549	1.69	0	8;18	168,549	0.39	0	8;18	8	0.44	122.78	123.61	168,549	0.19	0	8;18	9	0.63	$BV_1T_{IP}F_{EF}$
35	D/1.5			25	130,011	0.19	0	8;24	130,011	0.06	0	8;24	0	0.16	6.66	6.88	130,011	0.03	0	8;24	0	0.22	MIQCP
36	D			25	284,844	13,52	0	11,14;24	284,844	2.7	0	11;14;24	158	0.28	102.81	105.79	284,844	3.88	0	11;14;24	924	4.16	$BV_1T_{IP}F_{EF}$
37	D/1.2	4000	T	25	242,948	0.88	0	9;11;24	242,948	0.88	0	9;11;24	6	0.64	111.41	112.93	242,948	1.08	0	9;11;24	321	1.72	MIQCP
38	D/1.3			25	232,736	13.75	0	9;11;24	232,736	4.59	0	9;11;24	349	0.8	120	125,39	232,736	1.52	0	9;11;24	372	2.32	$BV_1T_{IP}F_{EF}$
39	D/1.4			25	210,573	0.05	0	9;24	210,573	0.05	0	9;24	0	0.44	122.78	123.27	210,573	0.05	0	9;24	0	0.49	MIQCP
40	D/1.5			25	202,214	8,67	0	9;24	202,214	0.016	0	9;24	0	0.16	6.66	6.84	202,214	0.16	0	9;24	1	0.32	$BV_1T_{IP}F_{EF}$

Table A.3: Comparison of $MIQCP$, $BV_1T_{IP}F_{MF}$ and $BV_2T_{IP}F_{MF}$ Results for 40 Nodes & 50 Nodes

					MIQCP				$BV_2T_{IP}F_{MF}$					TB (IP)			$BV_1T_{IP}F_{MF}$						
Ins	Demand	Cap	FC	Nodes	Obj	CPU	R.Gap	Hubs	Obj	CPU	R.Gap	Hubs	Fcut	T(i)	T(ij)	Tot(CPU)	Obj	CPU	R.Gap	Hubs	F.cut	Tot (CPU)	Best Method
41	D			40	254,209	7205.5	0.01	6;17;22;28;29	254,001	2097.23	0	6;17;22;28;37	4,869	1.22	362.41	2460.9	254,209	7200.03	0.006	6;17;22;28;29	30,797	7201.25	$BV_2T_{IP}F_{EF}$
42	D/1.2	3000	L	40	214,462	7200.3	0.02	14;15;29	214,462	150.92	0	6;22;25;28	325	1.22	386.8	538.94	215,025	7200.03	0.01	11;22;28;35	51,028	7201.25	$BV_2T_{IP}F_{EF}$
43	D/1.3			40	203,248	7057.3	0	6;22;25;28	203,248	802.58	0	6;22;25;28	8,841	1.17	421.33	1,225.08	203,804	7200.11	0.01	11;22;29;35	43,834	7201.28	$BV_2T_{IP}F_{EF}$
44	D/1.4			40	187,990	446.41	0	10;22;28	187,990	82.94	0	10; 22;28	862	1.14	457.52	541.6	187,990	5797.06	0	10;22;28	26,212	5798.2	MIQCP
45	D/1.5			40	175,597	67.34	0	14;25;29	175,597	21.38	0	14;25;29	174	1.95	624.27	647.6	175,597	7200.06	0.02	14;25;29	40,269	7202.01	MIQCP
46	D			40	353,868	7205.8	0.08	6;14;19;22;35	345,622	7200.08	0.03	6;14;19;22;35	608	1.22	362.41	7,563.7	349,249	7200.25	0.12	14;19;22;35;38	77,000	7201.47	$BV_2T_{IP}F_{EF}$
47	D/1.2	3000	T	40	288,997	7214.7	0.05	14;19;22;38	288,997	1165.05	0	14;19;22;38	271	1.22	386.8	1,553.07	291,639	7200.11	0.1	14;19;22;35	42,744	7201.33	$BV_2T_{IP}F_{EF}$
48	D/1.3			40	274,866	7205.1	0.06	1;14;22;35	275,337	7200.25	0.06	1;14;22;35	43,508	1.17	421.33	7,623	277,843	7200.05	0.14	6;19;22;35	42,343	7201.22	MIQCP
49	D/1.4			40	242,629	7200.1	0.002	14;22;35	242,629	7.59	0	14;22;35	66	1.14	457.52	466.25	243,781	7200.16	0.09	14;19;38	40,332	7201.3	$BV_2T_{IP}F_{EF}$
50	D/1.5			40	231,000	7202.7	0	14;19;22	231,000	32.25	0	14;19;22	249	1.95	624.27	658.47	231,739	7200.08	0.09	14;19;22	42,482	7202.03	$BV_2T_{IP}F_{EF}$
51	D			40	232,679	98.98	0	11;22;28;35	232,679	12.05	0	11;22;28;35	684	1.11	354.88	368.04	232,679	48.8	0	11;22;28;35	2,645	49.91	$BV_1T_{IP}F_{EF}$
52	D/1.2	4000	L	40	198,607	44.98	0	11;22;38	198,607	4.27	0	11;22;28	148	1.41	379.03	384.71	198,607	29.55	0	11;22;28	1,665	30.96	$BV_1T_{IP}F_{EF}$
53	D/1.3			40	185,104	51.13	0	11;22;28	185,104	20.2	0	11;22;28	1,320	1.13	470.25	491.58	185,104	24.58	0	11;22;28	1,392	25.71	$BV_1T_{IP}F_{EF}$
54	D/1.4			40	171,534	14.22	0	14;28	171,534	1.64	0	14;28	66	2.63	473	477.27	171,534	1.95	0	14;28	86	4.58	$BV_1T_{IP}F_{EF}$
55	D/1.5			40	161,281	1.11	0	14;28	161,281	0.11	0	14;28	0	1	623.95	625.06	161,281	0.11	0	14;28	0	1.11	$BV_1T_{IP}F_{EF}$
56	D			40	307,782	7205.9	0.03	1;14;19;38	308,617	7200.03	0.03	1;14;19;38	44,471	1.11	354.88	7,556	317,953	7200.06	0.08	14;19;22;38	48,274	7201.17	MIQCP
57	D/1.2	4000	T	40	263,294	7205.2	0.04	14;22;35	261,551	10.98	0	14;19;38	28	1.41	379.03	391.4	261,699	7200.06	0.06	14;19;38	53,157	7201.47	$BV_2T_{IP}F_{EF}$
58	D/1.3			40	251,187	7200.1	0.06	14;19;38	248,652	7200.11	0.05	14;19;22	52,550	1.13	470.25	7,671.5	249,747	7200.23	0.07	14;19;38	44,407	7201.36	$BV_2T_{IP}F_{EF}$
59	D/1.4			40	217,699	7.27	0	14;38	217,699	1.69	0	14;38	10	2.63	473	477.3	217,699	140.97	0	14;38	837	143.6	MIQCP
60	D/1.5			40	207,989	26.09	0	14;19	207,989	42.67	0	14;19	352	1	623.95	667.6	207,989	61.16	0	14;19	356	62.16	MIQCP
61	D			50	238,958	2337.83	0	3;22;27;45;48	238,958	141.31	0	3;22;27;45;48	582	1.83	645.1	788.2	238,958	7200.16	0.015	3;22;37;45;48	49,073	7,202	$BV_2T_2F_1$
62	D/1.2	3000	L	50	205,671	415.719	0	3;27;33;48	205,671	17.75	0	3;27;33;48	71	2.79	764.1	784.62	207,272	7200.14	0.04	3;27;33;48	48,807	7,203	MIQCP
63	D/1.3			50	193,248	636.89	0	3;22;27;48	193,248	41.58	0	3;22;27;48	257	1.73	903.1	946.42	193,248	7200.05	0.02	3;22;27;48	39,892	7,202	MIQCP
64	D/1.4			50	180,876	323.45	0	17;22;48	180,876	30.97	0	17;22;48	125	1.64	756.9	789.53	185,861	7200.36	0.05	15;34;48	27,670	7,202	MIQCP
65	D/1.5			50	168,983	70.81	0	17;22;48	168,983	15.81	0	17;22;48	49	1.06	735.3	752.17	171,986	7200.08	0.05	17;22;48	29,874	7,201	MIQCP
66	D			50	369,636	7207.86	0.06	3;22;27;45;48	367,544	7200.08	0.003	3;20;22;27;48	12,887	1.83	645.1	7847.01	370,150	7200.3	0.18	3;21;27;45;48	65,388	7,202	$BV_1T_{IP}F_{EF}$
67	D/1.2	3000	T	50	303,995	7206.11	0.03	3;22;27;48	303,645	888.7	0	3;22;27;48	506	2.79	764.1	1655.59	310,311	7200.05	0.14	3;22;27;48	53,116	7,201	$BV_2T_2F_1$
68	D/1.3			50	290,846	7208.38	0.06	3;21;27;48	287,790	7200.39	0.05	3;21;27;48	20,928	1.73	903.1	8105.2	312,562	7200.06	0.21	17;27;41;48	43,279	7,201	$BV_1T_{IP}F_{EF}$
69	D/1.4			50	260,197	2420.75	0	17;21;48	261,570	7200.13	0.02	21;27;48	27,931	1.64	756.9	7958.7	266,497	7200.05	0.14	17;24;48	37,999	7,201	MIQCP
70	D/1.5			50	246,105	3338.11	0	17;21;48	246,105	1777.09	0	17;21;48	10,030	1.06	735.3	2513.5	255,736	7200.33	0.15	17;24;48	36,704	7,201	$BV_2T_2F_1$
71	D			50	223,098	70.25	0	3;27;32;48	223,098	13.58	0	3;27;32;48	40	1.64	704.8	720.05	223,098	370.25	0	3;27;32;48	8,085	371.89	MIQCP
72	D/1.2	4000	L	50	192,416	39.69	0	17;22;48	192,416	13.89	0	17;22;48	46	0.95	792	806.87	192,416	1275.33	0	17;22;48	16,436	1,276.28	MIQCP
73	D/1.3			50	180,760	29.56	0	17;22;48	180,760	3.75	0	17;22;48	8	1.63	997.4	1002.8	180,760	117.97	0	17;22;48	1,951	119.6	MIQCP
74	D/1.4			50	171,012	120.41	0	15;48	171,012	199.63	0	15;48	742	1.98	828.2	1029.8	171,012	66.84	0	15;48	955	68.82	$BV_1T_{IP}F_{EF}$
75	D/1.5			50	159,790	10.47	0	15;48	159,790	8.06	0	15;48	19	1.97	910.6	920.6	159,790	2.77	0	15;48	17	4.74	$BV_1T_{IP}F_{EF}$
76	D			50	321,400	7205.33	0.05	3,21;27;48	320,478	7200.05	0.03	3;21;27;48	25,307	1.64	704.8	7906.5	333,864	7200.05	0.13	17;21;27;48	52,264	7,202	$BV_2T_2F_1$
77	D/1.2	4000	T	50	269,261	7204.02	0.04	17;21;48	268,726	349.23	0	3;27;48	1,030	0.95	792	1142.2	268,971	7200.09	0.07	17;21;48	46,095	7,201	$BV_2T_2F_1$
78	D/1.3			50	256,362	7204.92	0.06	17;21;48	256,325	7200.42	0.06	17;21;48	32,396	1.63	997.4	8199.48	256,325	7200.3	0.07	17;21;48	30,814	7,202	$BV_1T_{IP}F_{EF}$
79	D/1.4			50	226,257	138.73	0	17;48	226,257	106.75	0	17;48	396	1.98	828.2	936.93	226,257	489.8	0	17;48	2,132	491.78	MIQCP
80	D/1.5			50	214,552	21.02	0	17;48	214,552	189.45	0	17;48	676	1.97	910.6	1102.0	214,552	114.7	0	17;48	500	116.67	MIQCP

Table A.4: Results of $BV_1T_{LP}F_{EF}$ for 40 nodes

Instance	Demand	Cap	FC	Obj	CPU	B&B Nodes	R.Gap	Hubs	F.Cut	$BV_1T_{LP}F_{EF}$
1	D	4000	L	232,679	40.85	5,271	0	11;22;28;35	2,425	41.24
2	D/1.2			198,607	16.36	2,136	0	11;22;28	979	16.29
3	D/1.3			185,104	25.73	2,276	0	11;22;28	1,429	26.11
4	D/1.4			171,534	1.27	22	0	14;28	59	1.65
5	D/1.5			161,287	0.08	0	0	14;28	0	0.5
6	D	4000	T	315,864	7200,08	404,8	0.07	14;22;25;38	47,5	7,200.1
7	D/1.2			261,551	7,200,09	89,199	0.06	14;19;38	48,731	7,200.61
8	D/1.3			249,747	7,200,22	69,201	0.07	14;19;38	38145	7,200.67
9	D/1.4			217,699	11,64	761	0	14;38	724	12.16
10	D/1.5			207,989	5,09	315	0	14;19	299	5.53
11	D	3000	L	254,001	7,200.11	523,443	0.005	6;17;22;28;37	30,737	7,200.72
12	D/1.2			214,742	7,200.05	205,547	0.014	11;22;25;28	47,643	7,200.65
13	D/1.3			204,093	7,200.08	179,351	0.017	6;22;25;28	38,427	7,200.52
14	D/1.4			187,990	4,299.98	360,048	0	10;22;28	27,151	4,300.53
15	D/1.5			175,597	7,200.06	291,389	0.03	14;25;29	40,720	7,200.44
16	D	3000	T	355,784	7,200.25	181,621	0.13	14;19;22;35;38	70,159	7,200.83
17	D/1.2			290,029	7,200.08	163,27	0.1	14;19;22;35	45,069	7,200.53
18	D/1.3			277,056	7,200.06	271,929	0.13	14;19;22;35	52,422	7,200.47
19	D/1.4			242,629	7,200.09	279,821	0.08	14;22;35	53,156	7,200.5
20	D/1.5			236,297	7,200.11	77,618	0.11	14;19;22	53,113	7,200.52

Table A.5: Results of $BV_1T_{LP}F_{EF}$ for 50 Nodes instances

Instance	Demand	Cap	FC	Obj	CPU	B&B Nodes	R.Gap	Hubs	F.Cut	$BV_1T_{LP}F_{EF}$
1	D	4000	L	223,098	722.53	30,342	0	3;27;32;48	11,248	723.3
2	D/1.2			192,416	248.203	10,441	0	17;22;48	4,307	248.73
3	D/1.3			180,760	65.05	1891	0	17;22;48	2,056	65.66
4	D/1.4			171,012	26.81	411	0	15;48	780	27.2
5	D/1.5			159,790	1.55	17	0	15;48	19	1.8
6	D	4000	T	339,017	7,200.11	123,329	0.14	17;21;27;48	49,921	7,200.88
7	D/1.2			272,955	7,200.09	91,656	0.09	17;22;48	42,019	7,200.72
8	D/1.3			255,965	7,200.2	47,405	0.07	3;27;48	47,750	7,200.81
9	D/1.4			226,257	1,227.27	5,990	0	17;48	7,019	1,227.66
10	D/1.5			214,552	21.69	236	0	17,48	620	21.94
11	D	3000	L	238,958	7,200.06	271,557	0.015	3;22;27;45;48	44,239	7,200.84
12	D/1.2			208,783	7200.11	77,793	0.05	3;27;45;48	45,359	7,200.63
13	D/1.3			195,037	7,200.41	52,506	0.03	3;22;27;48	31,958	7,201.04
14	D/1.4			185,861	7,200.16	201,917	0.05	15;34;48	21,373	7,200.55
15	D/1.5			168,983	7,200.25	281,031	0.03	17;22;48	41,328	7,200.5
16	D	3000	T	386,566	7,200.08	130,167	0.21	17;21;27;45,48	59,315	7,200.86
17	D/1.2			317,743	7,200.08	89,664	0.16	3;27;45,48	46,592	7,200.6
18	D/1.3			321,061	7,200.05	98,990	0.23	17;21,26;48	41,413	7,200.68
19	D/1.4			266,045	7,200.11	115,445	0.13	17;24,48	52,069	7,200.5
20	D/1.5			253,778	7,200.08	146,261	0.13	17,22,48	51,259	7,200.33

Table A.6: Results of $BV_2T_{IP}F_{EF}$ for 40 nodes instances

Ins	Demand	Cap	FC	Obj	CPU	B&B Nodes	R.Gap	Hubs	F.Cut	$BV_2T_{LP}F_{EF}$
1	D	4000	L	232,679	36.08	4,368	0	11;22;28;35	2,237	37.19
2	D/1.2			198,607	59.27	13,619	0	11;22;28	2,635	60.68
3	D/1.3			185,104	28.34	4,303	0	11;22;28	1,908	29.47
4	D/1.4			171,534	1.7	26	0	14;28	86	4.33
5	D/1.5			161,281	0.09	0	0	14;28	0	1.09
6	D	4000	T	317,359	7,200.1	364,837	0.07	14;22;35;38	47,768	7,201.6
7	D/1.2			261,551	7,200.3	113,144	0.06	14;19;38	61,224	7,201.3
8	D/1.3			250,137	7,200.1	108,567	0.07	14;19;38	59,974	7,201.1
9	D/1.4			217,699	12.8	299	0	14;38	845	13.5
10	D/1.5			207,989	5.8	414	0	14;19	356	7.6
11	D	3000	L	254,001	7,200.0	502,018	0.005	6;17;22;28;37	31,872	7,202.0
12	D/1.2			214,462	7,200.1	256,978	0.013	6;22;25;28	49,907	7,200.8
13	D/1.3			203,248	4,764.1	162,629	0	6;22;25;28	42,552	4,765.0
14	D/1.4			187,990	2,718.4	290,834	0	10;22;28	19,638	2,719.5
15	D/1.5			175,597	7,200.1	307,300	0.013	14;25;29	50,698	7,203.4
16	D	3000	T	351,518	7,200.1	161,312	0.129	14;19;22;35;38	71,872	7,201.3
17	D/1.2			289,186	7,200.1	307,097	0.09	14;19;22;38	40,955	7,201.3
18	D/1.3			286,974	7,200.1	289,267	0.16	14;19;22;38	57,815	7,201.2
19	D/1.4			244,190	7,200.1	387,418	0.08	14;19;38	48,535	7,201.2
20	D/1.5			233,922	7,200.1	156,051	0.1	14;19;22	61,550	7,202.1

CHAPTER 2

APPENDIX B

Table 2.1: Detailed results of MISOCP and BDTA1 for WLANDP1 (from Pr1 to Pr5)

	MISOCP				**BDTA1**							
Pr	**CPU**	**Nodes**	**gap**	**Status**	**CPU**	**Nodes**	**Status**	**APs**	**callbacks**	**no of feas**	**no of opt**	**% feas**
1 (1)	19.9	25	0	Opt	12.7	139	Opt	2	126	106	20	85
1 (2)	288.5	5068	0	Opt	85.7	365	Opt	2	406	316	90	78
1 (3)	111.1	2495	0	Opt	338.7	1142	Opt	2	886	670	216	76
1 (4)	98.5	2563	0	Opt	335	440	Opt	2	1193	915	278	77
1 (5)	56.1	637	0	Opt	231.6	157	Opt	2	1367	1067	300	79
2 (1)	534.6	39771	0	Opt	153.1	469	Opt	3	499	422	77	85
2 (2)	3600.1	394635	11.8	Feas	2456.8	2555	Opt	3	1970	1748	222	89
2 (3)	2462.1	252066	0	Opt	2384.2	5346	Opt	3	2060	1313	747	64
2 (4)	435.1	40181	0	Opt	2471.7	4459	Opt	3	2144	1983	161	93
2 (5)	160.4	7873	0	Opt	402	1027	Opt	3	795	733	62	93
3 (1)	3600.1	83183	9.1	Feas	3601.8	4543	ATL	4	2603	2452	151	95
3 (2)	1873.9	254304	0	Opt	3602.2	3047	ATL	4	2216	2135	81	97
3 (3)	3600.1	92816	4.3	Feas	3602.5	4644	ATL	4	2135	1796	339	85
3 (4)	495.8	54400	0	Opt	3601.6	2098	ATL	4	2521	2479	42	99
3 (5)	3260.4	433558	0	Opt	3600.6	4188	ATL	4	2170	2122	48	98
4 (1)	3257.9	328738	0	Opt	369.8	1321	Opt	3	765	612	153	80
4 (2)	961.6	82901	0	Opt	3602.3	3545	ATL	3	2671	2196	475	83
4 (3)	1519.3	69818	0	Opt	3601.7	4033	ATL	3	2565	1859	706	73
4 (4)	312.5	25127	0	Opt	3603	5058	ATL	3	2589	2387	202	93
4 (5)	516.2	56158	0	Opt	1108,9	2579	Opt	3	1398	1221	177	88
5 (1)	3275.7	250566	0	Opt	3602.6	5800	ATL	4	2391	2069	322	87
5 (2)	1599.4	213649	0	Opt	3600.2	4359	ATL	4	2337	2145	192	92
5 (3)	484.9	49450	0	Opt	3601.4	4351	ATL	4	2283	1992	291	88
5 (4)	3256.8	472892	0	Opt	3601.8	2723	ATL	4	2407	2342	65	98
5 (5)	799.5	96631	0	Opt	3601.3	3362	ATL	4	2515	2413	102	96

Table 2.2: Detailed results of MISOCP and BDTA1 for WLANDP1 (from Pr6 to Pr10)

	MISOCP				**BDTA1**							
Pr	**CPU**	**Nodes**	**gap**	**Status**	**CPU**	**Nodes**	**Status**	**APs**	**callbacks**	**no of feas**	**no of opt**	**% feas**
6 (1)	3266.1	342159	0	Opt	3601.5	1489	ATL	6	2338	2304	34	98.6
6 (2)	3277.8	479018	0	Opt	3602.1	1046	ATL	6	2321	2312	9	99.7
6 (3)	3600.1	36205	3,04	Feas	3620.9	780	ATL	6	1987	1975	12	99.4
6 (4)	3363.9	523503	0	Opt	-	-	NS	-	-	-	-	-
6 (5)	1438.1	196709	0	Opt	3602.7	1294	ATL	6	2360	2342	18	99.3
7 (1)	206.2	1372	0	Opt	3604.3	4615	ATL	2	1962	1408	554	72
7 (2)	310	957	0	Opt	232.4	685	Opt	2	480	386	94	81
7 (3)	335.2	775	0	Opt	306.9	877	Opt	2	559	425	134	77
7 (4)	67.9	848	0	Opt	458.6	1184	Opt	2	760	641	119	85
7 (5)	62.2	849	0	Opt	95.1	671	Opt	2	336	265	71	79
8 (1)	3600.1	119286	9,15	Feas	3601.9	3224	ATL	3	1937	1633	340	85
8 (2)	3277.9	142681	0	Opt	3601.2	2790	ATL	3	2095	2046	49	98
8 (3)	3600.1	139021	12,32	Feas	3600.5	3894	ATL	3	2052	1708	344	84
8 (4)	3600.2	99380	14,44	Feas	3600.7	3054	ATL	3	1925	1879	46	98
8 (5)	3600.1	206456	7,41	Feas	3602.3	206456	ATL	3	1951	1876	75	97
9 (1)	3600.2	180365	9.2	Feas	3603.9	2126	ATL	4	1828	1756	72	97
9 (2)	3268.1	125121	9.4	Opt	3601.4	-	ATL	4	1870	1839	31	99
9 (3)	3274.5	124935	0	Opt	3606.5	2748	ATL	4	1937	1857	80	96
9 (4)	1155.8	70551	0	Opt	3602.9	2704	ATL	4	1913	1889	24	99
9 (5)	2926.5	214436	9.4	Opt	3604.2	2238	ATL	4	1995	1945	50	98
10 (1)	3600.2	68558	6,65	Feas	3600.2	3332	ATL	3	2054	1684	370	82
10 (2)	2359.3	76918	0	Opt	3603.8	2297	ATL	3	1919	1650	269	86
10 (3)	1413.6	43809	0	Opt	3601.5	5484	ATL	3	1858	1466	392	79
10 (4)	1388	59289	0	Opt	3600.9	1921	ATL	3	1921	1762	159	92
10 (5)	1813.4	101265	0	Opt	3603.5	4034	ATL	3	1855	1643	212	89

Table 2.3: Detailed results of MISOCP and BDTA1 for WLANDP1 (from Pr11 to Pr15)

	MISOCP				**BDTA1**							
Pr	**CPU**	**Nodes**	**gap**	**Status**	**CPU**	**Nodes**	**Status**	**APs**	**callbacks**	**no of feas**	**no of opt**	**% feas**
11 (1)	728.2	43971	0	Opt	3609.4	2557	ATL	4	1852	1781	71	97
11 (2)	209.7	11963	0	Opt	3600.4	2106	ATL	4	2042	1990	52	98
11 (3)	177.4	13254	0	Opt	3602	2344	ATL	4	1999	1916	83	96
11 (4)	120.9	5055	0	Opt	3603.1	2453	ATL	4	1850	1760	90	96
11 (5)	483.5	39879	0	Opt	3603.6	2137	ATL	4	1964	1889	75	97
12 (1)	3286.7	155369	0	Opt	3600.8	1114	ATL	6	2001	1986	15	99.3
12 (2)	3249.0	316418	0	Opt	-	-	NS	-	-	-	-	-
12 (3)	3600.2	34402	1.6	Feas	-	-	NS	-	-	-	-	-
12 (4)	3305.6	241153	0	Opt	-	-	NS	-	-	-	-	-
12 (5)	1474.0	131146	5	Opt	-	-	NS	-	-	-	-	-
13 (1)	0.95	0	0	Opt	1.8	53	Opt	1	40	22	18	55
13 (2)	1.1	0	0	Opt	0.8	1	Opt	1	47	27	20	58
13 (3)	1.2	0	0	Opt	7.2	74	Opt	1	113	67	46	60
13 (4)	0.97	0	0	Opt	0.8	1	Opt	1	118	69	49	59
13 (5)	1.31	0	0	Opt	8.1	46	Opt	1	167	104	63	63
14 (1)	42.1	2657	0	Opt	143.0	646	Opt	2	530	378	152	72
14 (2)	14.6	472	0	Opt	260.1	578	Opt	2	879	575	304	66
14 (3)	129.6	416	0	Opt	451.2	621	Opt	2	1286	895	391	70
14 (4)	13.0	15	0	Opt	348.8	229	Opt	2	1552	1111	441	72
14 (5)	33.2	982	0	Opt	704.5	592	Opt	2	1955	1439	516	74
15 (1)	87.2	8544	0	Opt	2586.9	6224	Opt	3	2331	2022	309	86.8
15 (2)	3600.2	3112	8.5	Feas	3600.1	3112	ATL	3	3604	2995	609	83.2
15 (3)	280.7	32505	0	Opt	3504.3	7298	Opt	3	2842	2095	747	73.8
15 (4)	756.5	95281	0	Opt	1745.0	910	Opt	3	3554	2743	811	77.2
15 (5)	324.7	24971	0	Opt	787.8	1823	Opt	3	1264	1013	251	80.2

Table 2.4: Detailed results of MISOCP and BDTA1 for WLANDP1 (from Pr16 to Pr20)

	MISOCP				BDTA1							
Pr	**CPU**	**Nodes**	**gap**	**Status**	**CPU**	**Nodes**	**Status**	**APs**	**callbacks**	**no of feas**	**no of opt**	**% feas**
16 (1)	27.6	1500	0	Opt	601.6	1899	Opt	2	1109	821	288	75
16 (2)	45.0	4658	0	Opt	1031	2427	Opt	2	1840	1202	638	66
16 (3)	13.5	19	0	Opt	586.9	2137	Opt	2	1072	844	228	79
16 (4)	23.7	1219	0	Opt	376.7	414	Opt	2	1372	1072	300	79
16 (5)	25.0	416	0	Opt	1961.8	2828	Opt	2	2346	1761	585	76
17 (1)	14.7	346	0	Opt	3600.1	3497	ATL	3	2609	2323	286	90
17 (2)	60.9	9382	0	Opt	3600.7	6464	ATL	3	2138	1482	656	70
17 (3)	190.0	30165	0	Opt	3600.3	6440	ATL	3	2846	2262	584	80
17 (4)	27.7	2689	0	Opt	2349.7	3770	Opt	3	2260	1922	338	86
17 (5)	243.9	36163	0	Opt	3600.6	6462	ATL	3	2788	2321	467	84
18 (1)	3600.1	231155	8.84	Feas	5460.1	1261	ATL	5	1625	1577	47	97.1
18 (2)	3600.1	195988	7.57	Feas	3600.2	1194	ATL	5	2594	2487	106	95.9
18 (3)	2088.4	91310	0	Opt	3601.4	3752	ATL	5	2478	2298	180	92.8
18 (4)	365	52520	0	Opt	3600.1	1159	ATL	5	2302	2292	10	99.6
18 (5)	3600.1	360393	5.07	Feas	3600.5	3253	ATL	5	2440	2341	99	96.0
19 (1)	1.9	0	0	Opt	0.3	0	Opt	1	4	2	2	50
19 (2)	2	0	0	Opt	27.4	501	Opt	1	177	114	63	64.5
19 (3)	2.1	0	0	Opt	2.2	1	Opt	1	183	118	65	64.5
19 (4)	2.2	0	0	Opt	12.8	53	Opt	1	221	132	89	59.8
19 (5)	2.5	0	0	Opt	12.8	14	Opt	1	251	155	96	61.8
20 (1)	23.4	47	0	Opt	38.5	215	Opt	2	201	177	24	88.1
20 (2)	23.4	43	0	Opt	1726.8	5133	Opt	2	1519	934	585	61.5
20 (3)	76.8	454	0	Opt	1007.3	307	Opt	2	1926	1305	621	67.8
20 (4)	97.1	758	0	Opt	199.9	538	Opt	2	498	416	82	83.6
20 (5)	20	151	0	Opt	124	154	Opt	2	638	530	108	83.1

Table 2.5: Detailed results of MISOCP and BDTA1 for WLANDP1 (from Pr21 to Pr24)

	MISOCP				BDTA1							
Pr	**CPU**	**Nodes**	**gap**	**Status**	**CPU**	**Nodes**	**Status**	**APs**	**callbacks**	**no of feas**	**no of opt**	**% feas**
21 (1)	3600.2	186364	0.004	Feas	2274	2312	Opt	3	1796	1703	93	94.9
21 (2)	3600.2	97284	9.95	Feas	3600.2	6790	ATL	3	2371	2047	324	86.4
21 (3)	2334.6	121780	0	Opt	2084.9	2021	Opt	3	1637	1512	125	92.4
21 (4)	530.7	32713	0	Opt	3600.6	2845	ATL	3	2452	2365	87	96.5
21 (5)	545.7	23796	0	Opt	649.1	1331	Opt	3	936	851	85	91.0
22 (1)	17.2	70	0	Opt	51.7	483	Opt	2	279	223	56	80.0
22 (2)	36.6	1857	0	Opt	3601.6	11973	ATL	2	2259	1075	1184	47.6
22 (3)	66.3	1447	0	Opt	31.0	127	Opt	2	205	177	28	86.4
22 (4)	11	25	0	Opt	164.5	686	Opt	2	507	407	100	80.3
22 (5)	11.6	93	0	Opt	20.4	115	Opt	2	153	124	29	81.1
23 (1)	322.9	34471	0	Opt	3600.4	4508	ATL	3	2259	2046	213	90.6
23 (2)	3600.1	175701	1.42	Feas	3600.1	9866	ATL	3	2377	1732	645	72.9
23 (3)	17.5	78	0	Opt	2659.4	3331	Opt	3	2115	1905	210	90.1
23 (4)	125.4	634	0	Opt	3601.9	3848	ATL	3	2343	2156	187	92.1
23 (5)	156.2	8434	0	Opt	2656.3	2071	Opt	3	1504	1340	164	89.1
24 (1)	3600.2	352242	0.004	Feas	3600	2470	ATL	5	2353	2334	19	99.2
24 (2)	3600.1	105695	5.82	Feas	3600.6	2139	ATL	5	1267	1191	76	94.1
24 (3)	3600.2	63954	4.6	Feas	3601.2	3096	ATL	5	2237	2105	132	94.1
24 (4)	3600.1	376348	0.0002	Feas	3601.3	1725	ATL	5	2308	2299	9	99.7
24 (5)	3604.6	306658	0.03	Feas	3836.4	900	ATL	5	626	600	26	95.9

Table 2.6: Detailed results of MISOCP, BDTA2 and BBC for WLANDP2 (from Pr1 to Pr8)

	MISOCP				**(BDTA2)**					**Branch and Benders Cut (BBC)**			
Pr	**CPU**	**Nodes**	**gap (%)**	**Status**	**CPU**	**Nodes**	**Status**	**Aps**	**Cuts**	**CPU**	**Nodes**	**Status**	**Cuts**
1 (1)	17.8	198	0	Opt	**3.8**	18	Opt	2	67	8.1	29	Opt	63
1 (2)	36.8	17	0	Opt	**12.4**	33	Opt	2	96	40.1	139	Opt	117
1 (3)	29.4	88	0	Opt	**26.1**	46	Opt	2	114	27.1	16	Opt	51
1 (4)	25.1	24	0	Opt	**19**	41	Opt	2	60	29.3	83	Opt	41
1 (5)	**19.8**	21	0	Opt	20.7	26	Opt	2	57	53.1	71	Opt	71
2 (1)	3600	254045	19.1	Feas	21	158	Opt	3	192	**20.3**	281	Opt	101
2 (2)	83.4	491	0	Opt	90.2	238	Opt	3	471	**54.6**	1060	Opt	215
2 (3)	3600	267335	8.9	Feas	**171**	229	Opt	3	778	188.9	1374	Opt	420
2 (4)	3600	309925	5.7	Feas	**527**	567	Opt	3	1324	1123.1	620	Opt	981
2 (5)	70	560	0	Opt	555	271	Opt	3	1732	**65.6**	632	Opt	186
3 (1)	3600	237628	16.4	Feas	328	689	Opt	4	842	**79.2**	4820	Opt	4820
3 (2)	**71.8**	630	0	Opt	1697	1413	OptTol	4	1859	557.9	67.43	OptTol	611
3 (3)	**651.8**	34522	0.1	Opt	958	1929	Opt	4	1162	1210.6	5115	Opt	978
3 (4)	**45.5**	278	0	Opt	3602	1394	ATL	4	3039	1356.2	3186	OptTol	799
3 (5)	1346	27301	0	Opt	2262	1906	OptTol	4	2166	**1262.2**	1583	Opt	1242
4 (1)	729	35336	0	Opt	**20.1**	158	Opt	3	192	31.1	598	Opt	112
4 (2)	**83.8**	355	0	Opt	90.7	471	Opt	3	471	441.4	438	Opt	564
4 (3)	3600	328460	4.2	Feas	**169**	229	Opt	3	778	445.2	920	Opt	703
4 (4)	**58.5**	337	0	Opt	519	567	Opt	3	1324	828.7	284	OptTol	957
4 (5)	75.1	209	0	Opt	122	271	Opt	3	408	**57**	1559	Opt	139
5 (1)	**80.3**	2534	0	Opt	368	664	Opt	4	874	472.7	1582	Opt	577
5 (2)	**17.1**	30	0	Opt	1269	1413	OptTol	4	1891	2345.7	1164	OptTol	7292
5 (3)	**27.5**	420	0	Opt	648	1624	Opt	4	1191	180.7	4623	Opt	332
5 (4)	**14.1**	16	0	Opt	3605	1258	ATL	4	2939	1756.1	19055	OptTol	957
5 (5)	**16.2**	20	0	Opt	2232	1906	OptTol	4	2166	1011.5	9207	Opt	786
6 (1)	**24.6**	897	0	Opt	3601	1099	ATL	6	2910	3602.3	7379	ATL	1600
6 (2)	**32.6**	1096	0	Opt	3601	1204	ATL	6	2684	3610.9	53040	ATL	1594
6 (3)	1455	26434	0.1	Opt	3601	1494	ATL	6	2808	3600.1	89423	ATL	1422
6 (4)	**17.7**	279	0	Opt	-	-	NS	6	-	3603.3	29461	ATL	1534
6 (5)	**25.6**	585	0	Opt	3601	1148	ATL	6	2877	3607.9	4425	ATL	1473
7 (1)	37.5	87	0	Opt	**30.3**	0	Opt	2	183	176.2	5275	Opt	233
7 (2)	147.3	2312	0	Opt	**67.8**	0	Opt	2	338	1473.5	18331	OptTol	774
7 (3)	49.8	18	0	Opt	**34.8**	0	Opt	2	390	75.4	4272	Opt	118
7 (4)	161.5	1592	0	Opt	**97.1**	0	OptTol	2	511	1228.3	14439	OptTol	646
7 (5)	**62.2**	124	0	Opt	104	0	Opt	2	604	474.9	4841	OptTol	393
8 (1)	3600	141660	0	Feas	**254**	0	Opt	3	559	740.2	21690	OptTol	525
8 (2)	3600	149626	0	Feas	**1509**	0	Opt	3	1500	2884.7	15972	OptTol	827
8 (3)	2356	60084	0	Opt	-	-	NS	3	-	**1524.5**	90378	OptTol	606
8 (4)	**1184**	17887	0	Opt	2240	0	OptTol	3	1773	3610.4	7156	ATL	1163
8 (5)	165.1	352	0	Opt	**189**	0	Opt	3	506	3619.6	10843	ATL	1162

Table 2.7: Detailed results of MISOCP, ILS2 and BBC for GWLANP2 (from Pr9 to Pr16)

	MISOCP1				**Integer L-shaped (ILS2)**					**Branch and Benders Cut (BBC)**			
Pr	**CPU**	**Nodes**	**gap (%)**	**Status**	**CPU**	**Nodes**	**Status**	**Aps**	**Cuts**	**CPU**	**Nodes**	**Status**	**Cuts**
9 (1)	3600.2	45284	18.4	Feas	-	-	NS	4	-	924.7	6968	OptTol	670
9 (2)	**193.5**	866	0	Opt	3600.5	-	ATL	4	2175	0		NS	-
9 (3)	3600.1	116770	24.3	Feas	3600.5	-	ATL	4	2217	**281.7**	30259	OptTol	321
9 (4)	**150.3**	1295	0	Opt	3602.7	-	ATL	4	2117	-	-	NS	-
9 (5)	**199.3**	1477	0	Opt	-	-	NS	-	-	-	-	NS	-
10 (1)	3600.2	131185	22,1	Feas	249.7	503	Opt	3	559	**176.2**	5275	Opt	233
10 (2)	**254.6**	2235	0	Opt	1360.3	778	Opt	3	1500	1473.5	18331	OptTol	774
10 (3)	3175.5	109900	0,1	Opt	**46.8**	260	Opt	3	242	75.4	4272	Opt	118
10 (4)	**218.4**	1194	0	Opt	2171.3	2027	OptTol	3	1773	1228.3	14439	OptTol	646
10 (5)	**162.9**	492	0	Opt	174.1	482	Opt	3	506	474.9	4841	OptTol	393
11 (1)	**495.9**	21161	0.1	Opt	3600.8	2068	ATL	4	2187	740.2	21690	OptTol	525
11 (2)	**52.2**	247	0	Opt	3600.8	2322	ATL	4	2159	2884.7	15972	OptTol	827
11 (3)	**341**	6645	0	Opt	3602.4	2956	ATL	4	2298	1524.5	90378	OptTol	606
11 (4)	**59.9**	168	0	Opt	3601.7	2015	ATL	4	2246	3610.4	7156	ATL	1163
11 (5)	**79.6**	224	0	Opt	3600.7	1926	ATL	4	2267	3619.6	10843	ATL	1162
12 (1)	3446.4	56944	0,1	Opt	-	-	NS	6	-	-	-	NS	-
12 (2)	**252.1**	6225	0	Opt	-	-	NS	6	-	-		NS	-
12 (3)	**562.5**	4800	0.1	Opt	3600.8	-	ATL	6	-	3600.1	6500	ATL	1130
12 (4)	**125.8**	1783	0	Opt	-	-	NS	6	-	-	-	NS	-
12 (5)	**163.5**	5640	0	Opt	3602.5	-	ATL	6	840	-	-	NS	-
13 (1)	1.3	0	0	Opt	**0.1**	0	Opt	1	4	0.4	1	Opt	6
13 (2)	1.5	0	0	Opt	**0.3**	0	Opt	1	8	0.5	0	Opt	7
13 (3)	1.5	0	0	Opt	**0.4**	0	Opt	1	7	0,4	0	Opt	4
13 (4)	1.3	0	0	Opt	**0.3**	0	Opt	1	4	0.3	0	Opt	3
13 (5)	1.9	0	0	Opt	**0.8**	0	Opt	1	8	1.3	1	Opt	10
14 (1)	11.2	122	0	Opt	7.6	43	Opt	2	117	**5.3**	126	Opt	42
14 (2)	15.2	79	0	Opt	**6.6**	33	Opt	2	54	11.2	40	Opt	58
14 (3)	24.7	319	0	Opt	**11.3**	38	Opt	2	75	93.5	162	Opt	207
14 (4)	**17.7**	319	0	Opt	24.4	50	Opt	2	75	21.7	25	Opt	31
14 (5)	**12**	55	0	Opt	22.6	30	Opt	2	77	44.8	27	Opt	53
15 (1)	**12.4**	37	0	Opt	60.1	215	Opt	3	396	69.9	7593	Opt	166
15 (2)	3600.1	317983	19.7	Feas	**83.8**	428	Opt	3	787	118.1	567	Opt	139
15 (3)	**372.4**	37073	0	Opt	111.9	400	Opt	3	561	259.4	780	OptTol	246
15 (4)	**27.7**	301	0	Opt	187.5	733	Opt	3	727	127.5	439	Opt	115
15 (5)	665.0	61507	0	Opt	95.4	462	Opt	3	95.4	**116**	1569	Opt	129
16 (1)	9.1	10	0	Opt	7.4	43	Opt	2	117	**7.2**	96	Opt	43
16 (2)	13	166	0	Opt	6.3	33	Opt	2	54	**5.8**	50	Opt	31
16 (3)	15.3	145	0	Opt	**13.3**	38	Opt	2	75	36.8	116	Opt	117
16 (4)	**9**	18	0	Opt	22.6	50	Opt	2	103	16.5	34	Opt	32
16 (5)	**9**	13	0	Opt	22.6	39	Opt	2	77	16.7	21	Opt	27

Table 2.8: Detailed results of MISOCP, ILS2 and BBC for GWLANP2 (from Pr17 to Pr24)

	MISOCP1				Integer L-shaped (ILS2)					Branch and Benders Cut (BBC)			
Pr	**CPU**	**Nodes**	**gap (%)**	**Status**	**CPU**	**Nodes**	**Status**	**Aps**	**Cuts**	**CPU**	**Nodes**	**Status**	**Cuts**
17 (1)	**10.5**	52	0	Opt	61.2	226	Opt	3	402	56.2	478	Opt	216
17 (2)	3600	207949	4.5	Feas	157.2	368	Opt	3	793	**100.9**	845	Opt	373
17 (3)	**15.6**	56	0	Opt	366.9	419	Opt	3	1268	362.9	1338	Opt	697
17 (4)	**8.9**	21	0	Opt	593.2	430	Opt	3	1816	645.6	849	Opt	1036
17 (5)	**38.2**	1694	0	Opt	497.8	370	Opt	3	2195	710.1	806	Opt	1323
18 (1)	**15.8**	83	0	Opt	3600.8	2094	ATL	5	3163	941.6	8918	Opt	957
18 (2)	3600	183813	11.4	Feas	3601	1394	ATL	5	3209	**1403**	5626	Opt	1522
18 (3)	3600	17541	15.1	Feas	3601.9	1471	ATL	5	330	**1094.6**	16836	Opt	1019
18 (4)	**40.7**	1070	0	Opt	3601.7	1745	ATL	5	3220	3600.8	1903	ATL	2042
18 (5)	3600	258836	7	Feas	3601.5	2535	ATL	5	2535	**2896**	48558	OptTol	1692
19 (1)	1.8	0	0	Opt	**0.2**	0	Opt	1	4	0.6	1	Opt	7
19 (2)	1.9	0	0	Opt	**0.4**	0	Opt	1	8	0.6	0	Opt	7
19 (3)	1.8	0	0	Opt	**0.4**	0	Opt	1	7	1.1	1	Opt	8
19 (4)	2	0	0	Opt	**0.3**	0	Opt	1	4	0.9	1	Opt	5
19 (5)	1.9	0	0	Opt	**0.9**	0	Opt	1	11	2.1	1	Opt	11
20 (1)	17.9	49	0	Opt	**10.1**	72	Opt	2	123	23.1	112	Opt	90
20 (2)	**21.2**	22	0	Opt	29	77	Opt	2	139	49.4	93	Opt	93
20 (3)	55.2	609	0	Opt	**54.2**	121	Opt	2	147	182.3	129	Opt	183
20 (4)	**19.2**	30	0	Opt	121.5	138	Opt	2	168	45.4	73	Opt	38
20 (5)	**23.9**	25	0	Opt	56.2	42	Opt	2	528	193.5	105	Opt	130
21 (1)	**27.7**	301	0	Opt	187.5	733	Opt	3	727	127.5	439	Opt	115
21 (2)	3600	168322	28.5	Feas	213.2	677	OptTol	3	658	**154.5**	1789	Opt	265
21 (3)	**215.7**	6125	0	Opt	1188.4	929	Opt	3	2062	619.2	3004	OptTol	745
21 (4)	**107.6**	743	0	Opt	795.9	1466	Opt	3	1269	183.2	2395	Opt	327
21 (5)	3600	172096	9.7	Feas	**455.0**	1122	Opt	3	920	73.8	650	Opt	204
22 (1)	14.1	10	0	Opt	**10.3**	72	Opt	2	123	29.1	127	Opt	102
22 (2)	**28.4**	77	0	Opt	29.5	77	Opt	2	262	55	363	Opt	206
22 (3)	**199.8**	6125	0	Opt	200.5	934	Opt	3	632	222.7	3004	OptTol	314
22 (4)	**107.6**	743	0	Opt	795.9	1466	Opt	3	1269	183.2	2395	Opt	327
22 (5)	**37.1**	74	0	Opt	50.9	42	Opt	2	696	115.8	93	Opt	513
23 (1)	**23.1**	52	0	Opt	413.2	910	Opt	3	898	253.5	902	Opt	401
23 (2)	3600	152060	4.9	Feas	**858.6**	626	OptTol	3	626	512.2	2654	Opt	701
23 (3)	**34.4**	160	0	Opt	220.2	996	Opt	3	640	1124.4	3103	Opt	795
23 (4)	**17.2**	29	0	Opt	1149.2	810	Opt	3	1618	1167.3	3664	Opt	1113
23 (5)	**20.8**	51	0	Opt	70	335	Opt	3	350	30.9	541	Opt	114
24 (1)	**30.9**	72	0	Opt	3600.4	2034	ATL	5	2546	69,9	7593	Opt	166
24 (2)	3600	151947	15.1	Feas	3602,6	1850	ATL	5	2607	**253.2**	1789	Opt	431
24 (3)	3600	34833	14	Feas	3600.8	2145	ATL	5	2488	**619.2**	3004	OptTol	745
24 (4)	**42.2**	282	0	Opt	3600.7	1136	ATL	5	2462	172	2395	Opt	327
24 (5)	**66.9**	1321	0	Opt	3602.8	2023	ATL	5	2536	263.9	650	Opt	531

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: GÜNDOĞDU, Emine
Nationality: Turkish (TC)
Date and Place of Birth: 06.12.1987, Konya
Marital Status: Single
Email: emineegundogdu@gmail.com

EDUCATION

Degree	Institution	Year of Graduation
M.S.	TOBB University of Economics and Technology, Industrial Engineering	2013
B.S.	Dokuz Eylül University, Industrial Engineering	2010
High School	Samsun Huriye Süer Anatolian High School	2005

PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
2014-1018	METU, Industrial Engineering Department	Research & Teaching Assistant
2011-2013	TOBB ETU,Industrial Engineering Department	Research & Teaching Assistant

PUBLICATIONS

E.Gundogdu, H. Gultekin. Scheduling in two-machine robotic cells with a self-buffered robot. *IIE Transactions, 2016* doi:10.1080/0740817X.2015.1047475